DTIC
SELECTE
DEC 30 1994
F

Implementation and Demonstration of a Multiple Model Adaptive Estimation Failure Detection System for the F-16

THESIS
Peter Keith Eide
Captain, USAF

AFIT/GE/ENG/94D-06

This document has been approved for public release and sale; its distribution is unlimited.

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

19941228 060

AFIT/GE/ENG/94D-06

DTIC
ELECTE
DEC 3 0 1994
S D
F

Implementation and Demonstration of a Multiple Model Adaptive Estimation Failure Detection System for the F-16

Accesion For
NTIS CRA&I ✓
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution /
Availability Codes
Dist | Avail and / or Special
A-1

THESIS
Peter Keith Eide
Captain, USAF

AFIT/GE/ENG/94D-06

DTIC QUALITY INSPECTED 2

Approved for public release; distribution unlimited

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.

AFIT/GE/ENG/94D-06

Implementation and Demonstration of a Multiple Model Adaptive Estimation Failure Detection System for the F-16

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Peter Keith Eide, B.S.E.E.
Captain, USAF

December 1994

Approved for public release; distribution unlimited

Acknowledgments

This space is reserved for all the people who've provided me direction, encouragement, and support. I first thank my instructors for showing sincere interest in my development both as a student and an engineer. Special thanks go to my advisor, Dr Maybeck, for providing a continual ray of sunshine during the "darker days" of this thesis research. Secondly, I wish to acknowledge the super group of friends with whom I've had the pleasure of learning as well as loafing. I really couldn't have run into a nicer crowd. Thirdly, I thank my parents, Brian and Mary, who were far away but near at heart. Nobody makes it this far in life without the love and support of family. Last, but certainly not least, I must thank from the bottom of my heart, my girlfriend Jennie. She has weathered all of the storms and celebrated all of the successes with equal patience and kindness. I have been truly blessed by her friendship.

One last thanks to all who cared. You know who you are!

Peter Keith Eide

Table of Contents

Page

Acknowledgments ii

List of Figures vii

List of Tables xii

Abstract xiii

1. Introduction 1
  1.1 Chapter Overview 1
  1.2 General 1
  1.3 Problem Statement 2
  1.4 Algorithm Overview 3
  1.5 Research Objectives 7
    1.5.1 MMAE Implementation 8
    1.5.2 Flight Envelope Research 8
  1.6 Research Questions 9
    1.6.1 Probability Convergence 9
    1.6.2 Flight Envelope Research 11
  1.7 Scope 11
  1.8 Limitations/Assumptions 12
  1.9 Report Format 13
  1.10 Chapter Summary 13

2. Background and History 14
  2.1 Chapter Overview 14
  2.2 Multiple Model Adaptive Estimation 14
    2.2.1 Algorithm Development (Theoretical) 15

Page

2.2.2 Algorithm Development (Experimental) 21

2.2.3 Dithering 26

2.2.4 Residual Monitoring 27

2.3 MMAE Research 28

2.3.1 Flight Control at the Air Force Institute of Technology 28

2.3.2 Other and Elsewhere 30

2.4 Multiple Model Adaptive Control 30

2.5 Chapter Summary 31

3. Methodology and Model Development 33

3.1 Chapter Overview 33

3.2 Truth Model - SRF 33

3.2.1 Equations of Motion 33

3.2.2 Flight Control System 34

3.2.3 Atmospheric Disturbances 37

3.2.4 Input/Output 40

3.2.5 Failure Insertion 41

3.3 Reduced Order Kalman Filter Model 41

3.3.1 Dynamics Model (A, B, Φ and B_d Matrices) 42

3.3.2 Measurement Model (C, D, and H Matrices) 49

3.3.3 Noises (R, G, and Q Matrices) 52

3.3.4 Failure Modeling 57

3.4 Chapter Summary 58

4. Results 59

4.1 Chapter Overview 59

4.2 Probability Convergence 59

4.2.1 Single Failures 59

4.2.2 Dual Failures 105

4.2.3 Time of Failure Insertion Variability Demonstration 112

4.3 Residual Monitoring 114

4.3.1 Characterization of Residual Monitoring Results 114

4.3.2 Answers to Research Questions 130

4.4 MMAC Demonstration 130

4.5 Flight Envelope Research 150

4.5.1 Characterization of Flight Envelope Results 150

4.5.2 Answers to Research Questions 162

4.6 Chapter Summary 163

5. Conclusions and Recommendations 164

5.1 Chapter Overview 164

5.2 Conclusions 164

5.2.1 MMAE is a Viable Failure Detection Algorithm for Real World Implementation 164

5.2.2 Algorithm Design Process is Application Oriented 165

5.2.3 Full Envelope Coverage will Require Filter/Trim Gain Scheduling 165

5.2.4 Purposeful Excitation Must Not be Too Violent 166

5.2.5 Detuning May Alleviate Need for Dither Scheduling 167

5.2.6 Results Reported Represent Worst Case Scenario 167

5.2.7 Flight Control System Works Against Effective Dither 168

5.2.8 Continuous Dither Excitation May be Unnecessary 168

5.2.9 Robustness to Sensor Failure a Capability Worth Advancing . . 169

5.3 Recommendations 170

5.3.1 Design and Implement Gain Scheduling System 170

5.3.2 Develop the BIT Scheme 170

5.3.3 Design the MMAC 171

Page

5.3.4 Others 171

5.4 Chapter Summary 171

Appendix A. Research **Φ**, $\mathbf{B}_d$ and $\mathbf{Q}_d$ 173

Appendix B. VISTA SRF Parameter Values 177

Appendix C. SRF Modification Table 179

Appendix D. Filter Generation Process - *Step-by-Step* 181

Appendix E. Dual Failure Summary Plots 183

Bibliography 196

Vita 198

List of Figures

Figure Page

1. Multiple Model Adaptive Estimator 4
2. Probability Convergence: an Example Plot 10
3. Exaggerated Discretization of Flight Envelope 12
4. Thesis application of the MMAE algorithm 20
5. Hierarchical Structure 24
6. Multiple Model Adaptive Controller 31
7. MMAE-based Controller 32
8. VISTA F-16 Functional Block Diagram 35
9. ISA Bode Analysis 47
10. Fully Functional Aircraft with Original **Q** 63
11. Simulated *Left Stabilator* Actuator Failure with Original **Q** 64
12. Simulated *Right Stabilator* Actuator Failure with Original **Q** 65
13. Simulated *Left Flaperon* Actuator Failure with Original **Q** 66
14. Simulated *Right Flaperon* Actuator Failure with Original **Q** 67
15. Simulated *Rudder* Actuator Failure with Original **Q** 68
16. Simulated *Forward Velocity* Sensor Failure with Original **Q** 69
17. Simulated *Angle of Attack* Sensor Failure with Original **Q** 70
18. Simulated *Pitch Rate* Sensor Failure with Original **Q** 71
19. Simulated *Normal Acceleration* Sensor Failure with Original **Q** 72
20. Simulated *Roll Rate* Sensor Failure with Original **Q** 73
21. Simulated *Yaw Rate* Sensor Failure with Original **Q** 74
22. Simulated *Lateral Acceleration* Sensor Failure with Original **Q** 75
23. Fully Functional Aircraft with Longitudinal Pseudonoise 78
24. Simulated *Left Stabilator* Actuator Failure with Longitudinal Pseudonoise 79
25. Simulated *Right Stabilator* Actuator Failure with Longitudinal Pseudonoise 80
26. Simulated *Left Flaperon* Actuator Failure with Longitudinal Pseudonoise 81

Figure Page

27. Simulated *Right Flaperon* Actuator Failure with Longitudinal Pseudonoise 82

28. Simulated *Rudder* Actuator Failure with Longitudinal Pseudonoise 83

29. Simulated *Forward Velocity* Sensor Failure with Longitudinal Pseudonoise 84

30. Simulated *Angle of Attack* Sensor Failure with Longitudinal Pseudonoise 85

31. Simulated *Pitch Rate* Sensor Failure with Longitudinal Pseudonoise 86

32. Simulated *Normal Acceleration* Sensor Failure with Longitudinal Pseudonoise . . . 87

33. Simulated *Roll Rate* Sensor Failure with Longitudinal Pseudonoise 88

34. Simulated *Yaw Rate* Sensor Failure with Longitudinal Pseudonoise 89

35. Simulated *Lateral Acceleration* Sensor Failure with Longitudinal Pseudonoise 90

36. Simulated *Forward Velocity* Sensor Failure with Longitudinal Pseudonoise and Detuning in **R** 95

37. Simulated *Left Stabilator* Actuator Failure with Longitudinal and Lateral Pseudonoise and Detuning in **R** 97

38. Simulated *Yaw Rate* Sensor Failure with Longitudinal and Lateral Pseudonoise and Detuning in **R** 98

39. Simulated *Rudder* Actuator Failure with Longitudinal and Lateral Pseudonoise and Detuning in **R** 99

40. Simulated *Right Stabilator* Actuator Failure with Additional Detuning for Actuator Uncertainty 102

41. Time of Insertion Variability Demonstration 113

42. Comparison of Fully Functional and Left Stabilator Filter Residuals: (Left Stabilator Actuator Failure Induced at 3.0 seconds) 118

43. Comparison of Fully Functional and Right Stabilator Filter Residuals: (Right Stabilator Actuator Failure Induced at 3.0 seconds) 119

44. Comparison of Fully Functional and Left Flaperon Filter Residuals: (Left Flaperon Actuator Failure Induced at 3.0 seconds) 120

45. Comparison of Fully Functional and Right Flaperon Filter Residuals: (Right Flaperon Actuator Failure Induced at 3.0 seconds) 121

46. Comparison of Fully Functional and Rudder Residuals: (Rudder Actuator Failure Induced at 3.0 seconds) 122

Figure Page

47. Comparison of Fully Functional and Forward Velocity Filter Residuals: (Forward Velocity Sensor Failure Induced at 3.0 seconds) 123

48. Comparison of Fully Functional and Angle of Attack Filter Residuals: (Angle of Attack Failure Induced at 3.0 seconds) 124

49. Comparison of Fully Functional and Pitch Rate Filter Residuals: (Pitch Rate Sensor Failure Induced at 3.0 seconds) 125

50. Comparison of Fully Functional and Normal Acceleration Filter Residuals: (Normal Acceleration Sensor Failure Induced at 3.0 seconds) 126

51. Comparison of Fully Functional and Roll Rate Filter Residuals: (Roll Rate Sensor Failure Induced at 3.0 seconds) 127

52. Comparison of Fully Functional and Yaw Rate Filter Residuals: (Yaw Rate Sensor Failure Induced at 3.0 seconds) 128

53. Comparison of Fully Functional and Lateral Acceleration Filter Residuals: (Lateral Acceleration Sensor Failure Induced at 3.0 seconds) 129

54. Aircraft State Trajectory: *Left Stabilator* Actuator Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) 132

55. Aircraft State Trajectory: *Right Stabilator* Actuator Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) 133

56. Aircraft State Trajectory: *Left Flaperon* Actuator Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) 134

57. Aircraft State Trajectory: *Right Flaperon* Actuator Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) 135

58. Aircraft State Trajectory: *Rudder* Actuator Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) 136

59. Aircraft State Trajectory: *Angle of Attack* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) 137

60. Aircraft State Trajectory: *Pitch Rate* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) 138

61. Aircraft State Trajectory: *Normal Acceleration* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) 139

Figure Page

62. Aircraft State Trajectory: *Roll Rate* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) 140

63. Aircraft State Trajectory: *Yaw Rate* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) 141

64. Aircraft State Trajectory: *Lateral Acceleration* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) 142

65. Aircraft State Trajectory: *Angle of Attack* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) **No MMAE Input** 144

66. Aircraft State Trajectory: *Pitch Rate* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) **No MMAE Input** 145

67. Aircraft State Trajectory: *Normal Acceleration* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) **No MMAE Input** 146

68. Aircraft State Trajectory: *Roll Rate* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) **No MMAE Input** 147

69. Aircraft State Trajectory: *Yaw Rate* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) **No MMAE Input** 148

70. Aircraft State Trajectory: *Lateral Acceleration* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) **No MMAE Input** 149

71. Envelope Radial Line 1: – Nominal, - - Env pt #8, ·· Env pt #2, -·- Env pt #21 154

72. Envelope Radial Line 2: – Nominal, - - Env pt #15, ·· Env pt #24 155

73. Envelope Radial Line 3: – Nominal, - - Env pt #20, ·· Env pt #23 156

74. Envelope Radial Line 4: – Nominal, - - Env pt #14, ·· Env pt #13, -·- Env pt #22 157

75. Envelope Radial Line 5: – Nominal, - - Env pt #12, ·· Env pt #9 158

76. Envelope Radial Line 6: – Nominal, - - Env pt #18 159

77. Envelope Radial Line 7: – Nominal, - - Env pt #11, ·· Env pt #7, -·- Env pt #3 160

78. Envelope Radial Line 8: – Nominal, - - Env pt #17, ·· Env pt #19 161

79. Left Stabilator Actuator Summay Plot - Secondary Failure Induced at 5.0 seconds 184

80. Right Stabilator Actuator Summay Plot - Secondary Failure Induced at 5.0 seconds 185

81. Left Flaperon Actuator Summay Plot - Secondary Failure Induced at 5.0 seconds 186

Figure Page

82. Right Flaperon Actuator Summay Plot - Secondary Failure Induced at 5.0 seconds 187

83. Rudder Actuator Summay Plot - Secondary Failure Induced at 5.7 seconds 188

84. Forward Velocity Sensor Summay Plot - Secondary Failure Induced at 5.7 seconds . 189

85. Angle of Attack Sensor Summay Plot - Secondary Failure Induced at 5.0 seconds . 190

86. Pitch Rate Sensor Summay Plot - Secondary Failure Induced at 5.0 seconds 191

87. Normal Acceleration Sensor Summay Plot - Secondary Failure Induced at 5.0 seconds 192

88. Roll Rate Sensor Summay Plot - Secondary Failure Induced at 5.0 seconds 193

89. Yaw Rate Sensor Summay Plot - Secondary Failure Induced at 5.0 seconds 194

90. Lateral Acceleration Sensor Summay Plot - Secondary Failure Induced at 5.0 seconds 195

List of Tables

Table Page

1. Aircraft State Vector . . . 43
2. Control Vector . . . 43
3. Augmented Aircraft State Vector . . . 48
4. New Control Vector . . . 48
5. Sensor Variables Vector . . . 50
6. Vector of White Noises . . . 54
7. Zero-Order White Noise Strengths . . . 54
8. Dual Failure Performance Summary Matrix: First Failure (row) vs Second Failure (column) . . . 107
9. Dual Failure Performance Summary Matrix Key . . . 107
10. Envelope Investigation Matrix: Altitude (row) vs Mach (column), $\star$ denotes nominal point . . . 151
11. SRF Parameter Values . . . 178
12. Modified/Created SRF Files . . . 180

AFIT/GE/ENG/94D-06

Abstract

A Multiple Model Adaptive Estimation (MMAE) algorithm is implemented with the fully nonlinear six-degree-of-motion, Simulation Rapid-Prototyping Facility (SRF) VISTA F-16 software simulation tool. The algorithm is demonstrated to be capable of identifying flight critical aircraft actuator and sensor failures at a low dynamic pressure (20,000 ft, .4 Mach). Research included single and dual complete failures. Tuning methods for accommodating model mismatch, including addition of discrete dynamics pseudonoise and continuous measurement pseudonoise, are discussed and demonstrated. Scalar residuals within each filter are also examined and characterized for possible use as an additional failure declaration voter. Robustness to sensor failures provided by MMAE-based control is also demonstrated. An investigation of algorithm performance off the nominal design conditions is accomplished as a first step towards full flight envelope coverage.

The algorithm is composed of a bank of Kalman filters modeled to match particular hypotheses of the real world. Each presumes a single failure in one of the flight critical actuators (left/right stabilators, left/right flaperon, rudder), or sensors (forward velocity, angle of attack, pitch rate, normal acceleration, roll rate, yaw rate, and lateral acceleration), and one presumes no failure. For dual failures, a hierarchical structure is used to keep the number of on-line filters to a minimum. This research advances the technology by testing algorithm performance against the most complete simulation model currently available.

Implementation and Demonstration of a Multiple Model Adaptive Estimation Failure Detection System for the F-16

1. Introduction

1.1 Chapter Overview

This thesis is an implementation of a Multiple Model Adaptive Estimation (MMAE), probability based, failure detection and isolation algorithm. This chapter introduces the concepts and states the goals to be accomplished. Specifically, this chapter begins with a short motivational section, 1.2, before moving into the problem statement of Section 1.3. Section 1.4 then offers a "no equations allowed", top-level introduction to the MMAE algorithm. The specific objectives and research questions to be addressed are stated in Sections 1.5 and 1.6. The overall scope is next in Section 1.7, followed by any significant limitations or assumptions in Section 1.8. For convenience, the rest of this document is laid out in Section 1.9, the report format. Finally, the chapter concludes with a summary.

1.2 General

Currently, modern fighter aircraft such as the F-16 rely on a digital Flight Control System (FCS) to maintain static and dynamic stability. More and more, the FCS is being called on to perform tasks which go beyond the limits of human ability. So it is crucial then that the FCS operate safely and effectively, not only when an aircraft is fully functional, but also in the face of critical failures. For the F-16 under consideration in this thesis, critical failure modes are defined to be loss of control surface actuators and failure of sensors which provide stabilizing feedback signals to the FCS. There has been considerable research performed involving various failure detection and isolation schemes (1,3,5,7–10,14–18,23–28,30,32,33,35,37). However, most schemes can only identify a failure, being able then to do nothing but perhaps trigger a degraded mode in the FCS and hope to limp home to safety. Multiple Model Adaptive Control (MMAC) is a scheme which features, in addition to survivability, the significant characteristic that it reconfigures the FCS to operate effectively *without* the failed actuators or sensors.

The MMAC algorithm was first developed as far back as the early seventies (21). Unfortunately at the time, there were significant practical limitations to its usefulness. The digital computer was not nearly as advanced as it is today. Memory was expensive and more importantly, parallel processing on a digital computer had yet to be developed. It wasn't until the digital computer technology explosion that this algorithm's application even became feasible.

While the goal of this line of research is an implementable MMAC algorithm, step-by-step evolution requires that we first implement and demonstrate its basic component, the Multiple Model Adaptive Estimator (MMAE). Figure 1 and Chapter 2's Figure 6 show that the MMAC is basically an extension of the MMAE. In ultimate pursuit of the MMAC, this thesis will demonstrate the MMAE algorithm. Although both algorithms and their architectures are relatively simple, they are highly nonlinear and thus elude any complete theoretical analysis. They do however lend themselves to computer modeling and simulation. This thesis performs such an analysis to characterize the algorithm's performance attributes.

1.3 Problem Statement

Previous research efforts into MMAC and MMAE algorithms have addressed in a step-by-step manner: feasibility of the flight control application, application to the F-16 in particular, modifications to the algorithm for improved performance, and application to an experimental flight test bed (currently awaiting funding). In all cases though, there were significant limiting factors which must be taken into account when interpreting results. Computer simulations used linearized, reduced order truth models in representing real world behavior, and the test bed aircraft is an unmanned research vehicle with a top speed of 170 $\frac{ft}{sec}$ (14, 24–28, 30, 32, 33). Research sponsors, Wright Laboratory, WL/FIGS-4, Wright-Patterson AFB wish to be able to make a more realistic assessment of multiple model algorithm capabilities by applying them to a more sophisticated simulation truth model of the F-16. Such a truth model is now available in the Simulation/Rapid Prototyping (SRF) VISTA F-16 program which resides in the Flight Dynamics Laboratory. The SRF VISTA F-16 is a fully nonlinear, highly complex, six-degree-of-freedom aircraft simulation. As a truth model, it is much more realistic and includes many nonlinear dynamical effects which just cannot be accounted for in a linearized model. This thesis, then, will implement the MMAE algorithm developed in past research into the SRF environment and compare the performance results. Specifically, the ability of the MMAE to declare hard single and dual failures and provide estimates of current aircraft state to the F-16's Block-40 FCS in both axes, at both low and high dynamic pressure, and most importantly, in a nonlinear environment, will be investigated. The low dynamic pressure scenario (.4 Mach, 20,000 ft) has been retained as a worst case scenario and to facilitate direct comparison with past research (26). As closely as possible, the linearized model research will be duplicated in the nonlinear environment. Single and dual hard failures will be injected into the aircraft sensors and actuators. The ability of the MMAE to identify such failures correctly and in a timely fashion will be investigated. Performance will be directly compared with past research (26). In addition to implementation and demonstration, sponsors wish to advance the state-of-the-art by working towards full envelope coverage. To that end and using the SRF simulation tool, we will characterize the performance degradation when operating away from the nominal design point in the flight envelope. The above topics are revisited in Sections 1.5 and 1.6. The next section provides a brief introduction to the MMAE algorithm.

1.4 Algorithm Overview

Figure 1 illustrates the MMAE under consideration. It features a bank of K steady-state Kalman filters operating in parallel, each corresponding to a particular hypothesized failure con-

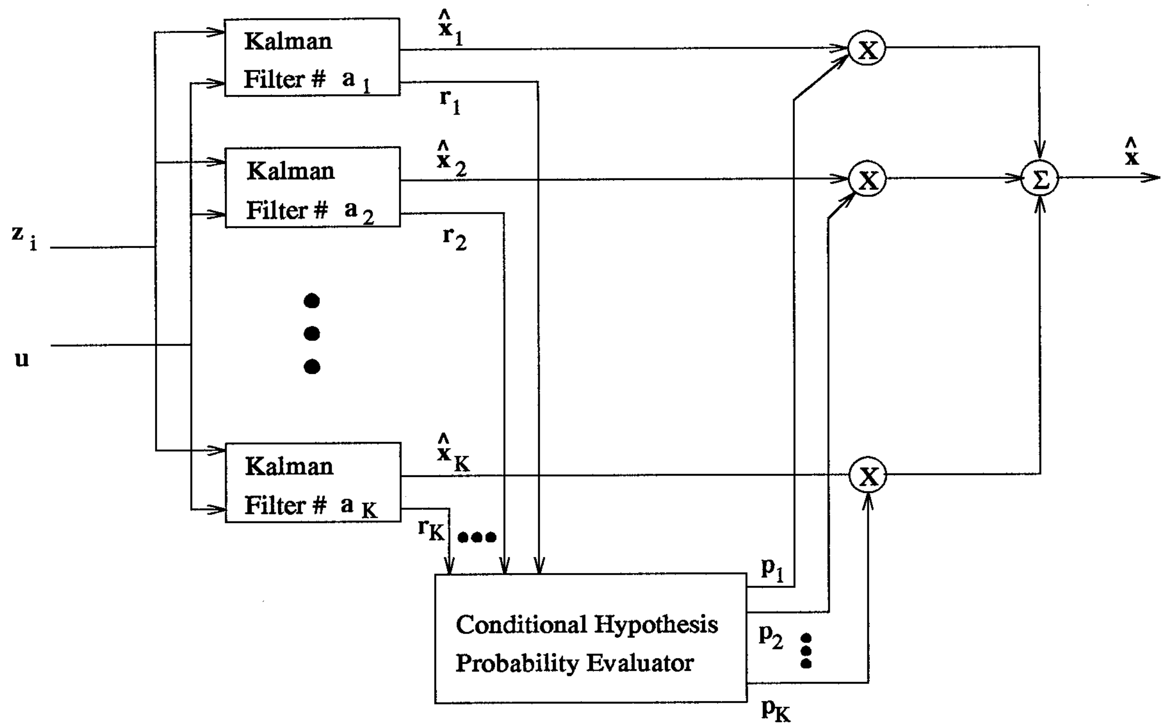


Figure 1. Multiple Model Adaptive Estimator

dition a_k, for $k = 1, 2, \ldots, K$. (The reader is assumed to have a good understanding of Kalman filters which are the basic unit within the MMAE (20–22).) Within the flight control application, the individual Kalman filters hypothesize no failures, or failed actuators, or failed sensors. Each filter has as its inputs, the latest vector of measurements $\mathbf{z}_i$ at sample time t_i, and current vector of control commands $\mathbf{u}$. Using knowledge of the inputs $\mathbf{u}$ and its internal model based on a_k, each filter produces an estimate of system state $\mathbf{x}$. In addition, each filter uses its state estimate to form a residual vector $\mathbf{r}_k$. In words, $\mathbf{r}_k$ is the difference between the measurement vector $\mathbf{z}_i$ and the filter's measurement vector prediction of that measurement before it arrives. If the k^{th} filter best represents the real world, or in other words, if its failure hypothesis is correct, its predicted measurement vector will be closest to the actual measurement vector. Its $\mathbf{r}_k$ will be small, on the order of its internally computed residual standard deviation. The rest of the filters in the bank will have made poor predictions of the measurement vector because their internal models don't match the real world. The filters are *mismatched*: their residuals are larger than their internally computed residual standard deviations.

The other prominent feature of Figure 1 is the hypothesis conditional probability evaluator. This portion of the code assigns a probability p_k to each filter based on the residual it has produced and its previous probability. p_k in words, is the probability that Kalman filter k is the best representation of the real world at the current time. Much more detail on the assignment of probabilities is provided in Chapter 2. How the probabilities are further used by the algorithm in selecting the correct filter was a subject of past research (30). Maximum a posteriori, which selects the state estimate generated by the filter with the highest probability only; Bayesian blending, which weights each state estimate by its probability and sums them; and modified Bayesian blending, which is the same as Bayesian but with lower limits imposed on the p_k's, were all evaluated. Modified Bayesian blending showed the best results and will be used in this thesis. A more precise description of modified Bayesian blending is given in Section 2.2.2.1 and Figure 1 also provides illustration.

An important piece of the algorithm which is not explicitly shown in Figure 1 is the dither. Dither is the term used for any purposeful, commanded "wiggling" of the control surface actuators for performance improvement. Consider aircraft steady level flight. Wings are level, there are no accelerations, no rotational rates, and the flaperons and rudder have no deflection. How is it possible to know that a rudder failure has occurred if it is not commanded to deflect and therefore does not *fail* to deflect? Stated more formally, the MMAE requires persistent state excitation, and further, of a sufficient magnitude to elicit large enough responses in order to show good failure detection performance. Continuous commanded input, or dither, is therefore necessarily introduced to the system at $\mathbf{u}$ if no inputs are generated by the pilot to command a maneuver. Past research has focused on the best dither signal to optimize performance (14, 26). Our research will use directly, results of this earlier research for the purpose of making direct comparisons. Much more information on dithering is provided in Chapter 2, Section 2.2.3, and throughout the report.

Multiple failures are handled by using a hierarchical structure. In a hierarchical approach to multiple failures, the architecture of Figure 1 is maintained, but the bank of elemental Kalman filters is redefinable. Upon identification of a first failure, a new bank of filters replaces the original one. This new bank contains filters which hypothesize two concurrent failures (all dual combinations of the first failure with the other possible single failures), a filter which hypothesizes the current single failure only, and one which hypothesizes no failures. The no-failure filter provides a "back door" to the original bank in the event the first failure is a false alarm, or simply disappears. Chapter 2, Figure 5 shows the hierarchical structure. More detail is provided in Chapter 2, Section 2.2.2.3.

A consideration which warrants discussion at this point is the soft failure and the MMAE's ability to identify it. An astute reader will have questioned the size of a bank of Kalman filters which cover all possible actuator and sensor failure modes. If soft, or degraded-mode, failures such as a 50% loss of a flaperon and a doubling of sensor noise in an accelerometer are allowed, there quickly becomes an infinite number of failures for which the algorithm needs to account. A major advantage of the MMAE structure and modified Bayesian blending is the ability to handle soft failures in actuators. If indeed the failure is a 50% loss of right flaperon, equally weighted state estimates from the filters based on the fully functional hypothesis and fully failed right flaperon hypothesis are blended together. Past research has demonstrated this effect nicely (26). Soft sensor failures are more difficult to handle and warrant further attention (26). This thesis will not address soft failures in the nonlinear environment.

Another point of interest not explicitly shown in Figure 1 is residual monitoring and its potential role in failure declaration. Recall that each filter produces a residual vector $\mathbf{r}_k$. Kalman filter theory tells us that the residuals of a well matched filter, or in our case, the filter with the correct failure hypothesis, ought to behave in a predictable way (20). Specifically, they ought to be zero-mean, white, Gaussian, and of predetermined covariance. Research has exploited this insight by looking directly at the individual scalar filter residuals to see which behave in this manner (26). Having found one indication of failure through the normal MMAE structure, the algorithm can then benefit from another, separate, indication of failure as corroboration. Residual monitoring is an example of what has been called an "additional voter" test. More detail on this technique is provided in Chapter 2, Section 2.2.4.

One final topic of discussion in this overview is MMAE-based control vs MMAC. As mentioned in Section 1.2, the ultimate goal is to achieve the robustness that a fully reconfigurable MMAC can offer. No separate controller is required. The algorithm is an estimator and controller all in one. How about applications such as the F-16 where a controller already exists? MMAC may be overkill. Certainly the MMAE algorithm could operate in parallel with the FCS and simply declare failures, but that would be under-utilizing its capabilities. A middle-ground possibility exists which makes the algorithm a feasible addition to existing control systems, and that is MMAE-based control illustrated in Figure 7. In this scheme, the system state vector created by the algorithm is used to reconstruct the vector of sensed values and then inputs it to the existing FCS in lieu of that system's usual sensor data. (Forward velocity is the only exception, however, because the FCS does not employ any direct measurement of forward velocity in its internal calculations.) Not only are the signals cleaned up, but it immediately makes the FCS robust to sensor failures. MMAE-based control has shown great promise and is in fact the form of control to be used in this thesis. This research will demonstrate the robustness to sensor failures.

1.5 Research Objectives

Previous research has investigated MMAE-based control cascaded with a FORTRAN model of the Block-40 FCS and a linear model of the F-16 (26). Other efforts have created a stand-alone flight test version of the MMAE algorithm (14). This thesis directly follows upon these efforts and applies the stand-alone MMAE algorithm to a stand-alone nonlinear F-16 simulator.

1.5.1 MMAE Implementation. The primary objective of this research is in the implementation of the MMAE algorithm into a fully nonlinear simulation environment. Sponsors in the Wright Laboratory, WL/FIGS-4, Wright-Patterson AFB have requested this work in the hopes of being able to make a more realistic assessment of multiple model algorithm capabilities. This will be accomplished by attaching the stand-alone MMAE algorithm developed in past research to the SRF VISTA F-16 aircraft simulation (9, 14). Care will be taken during implementation to recreate as closely as possible, the promising results of the past while advancing the state-of-the-art by noting any effects brought about from using the more sophisticated truth model. In addition to this primary thesis objective, a corollary objective is to characterize the effects on performance which result from moving the aircraft away from the nominal design point in the flight envelope. This is the first step toward achieving full envelope coverage.

1.5.2 Flight Envelope Research. MMAE research applied to the F-16 in the past has stayed exclusively at one point in the flight envelope. Since the algorithm is based on perturbation equations about a given trim condition, there is interest in observing the effects of moving off trim. In fact, studies performed on an unmanned research vehicle concluded that, in fact, algorithm performance does degrade off trim. The objective here then is to quantify this degradation across the entire flight envelope and determine the number of discrete versions of the algorithm required to cover the envelope. A scheduling scheme can then be designed which provides the required switching mechanism. Further, another goal is to tie the algorithm scheduling to the control gain scheduling which already takes place in the Block-40 FCS.

1.6 Research Questions

1.6.1 Probability Convergence. In this research, the most important measure of performance is how quickly the MMAE algorithm converges to the correct solution. When introducing failures, we would expect residuals in all mismatched elemental filters to be large, driving their probabilities to the minimum bound. As discussed further in Section 2.2.2.1, the minimum bound prevents "lock-out" or "lock-on" of any of the elemental filters. The probability associated with the elemental filter which matches true system failure status will then increase to the maximum. Figure 2 is an example of the type of plots which are used to display results in Chapter 4. It pertains to the case of a velocity sensor failure being induced at 3.0 seconds into the simulation. The p_k values are plotted for each of the elemental filters. The failure status hypothesis for each filter appears on the ordinate axis of each plot, and these are FF, fully functional; L ST, left stabilator actuator; R ST, right stabilator actuator; L FL, left flaperon actuator; R FL, right flaperon actuator; RUD, rudder actuator; VEL, velocity sensor; AOA, angle of attack sensor; PIT, pitch rate sensor; Az, normal acceleration sensor; ROL, roll rate sensor; YAW, yaw rate sensor; Ay, lateral acceleration sensor, failures. Figure 2 is drawn from past research and demonstrates excellent probability convergence (26).

This thesis research differs from prior research in that the truth model is fully nonlinear and uses higher order actuator models. The SRF VISTA F-16 model is more complete than the linearized model used in the past. Having made this change, some pertinent research questions include:

- Single Failures
  - *Do the elemental probabilities still converge to a solution?*
  - *Is the solution the correct one?*

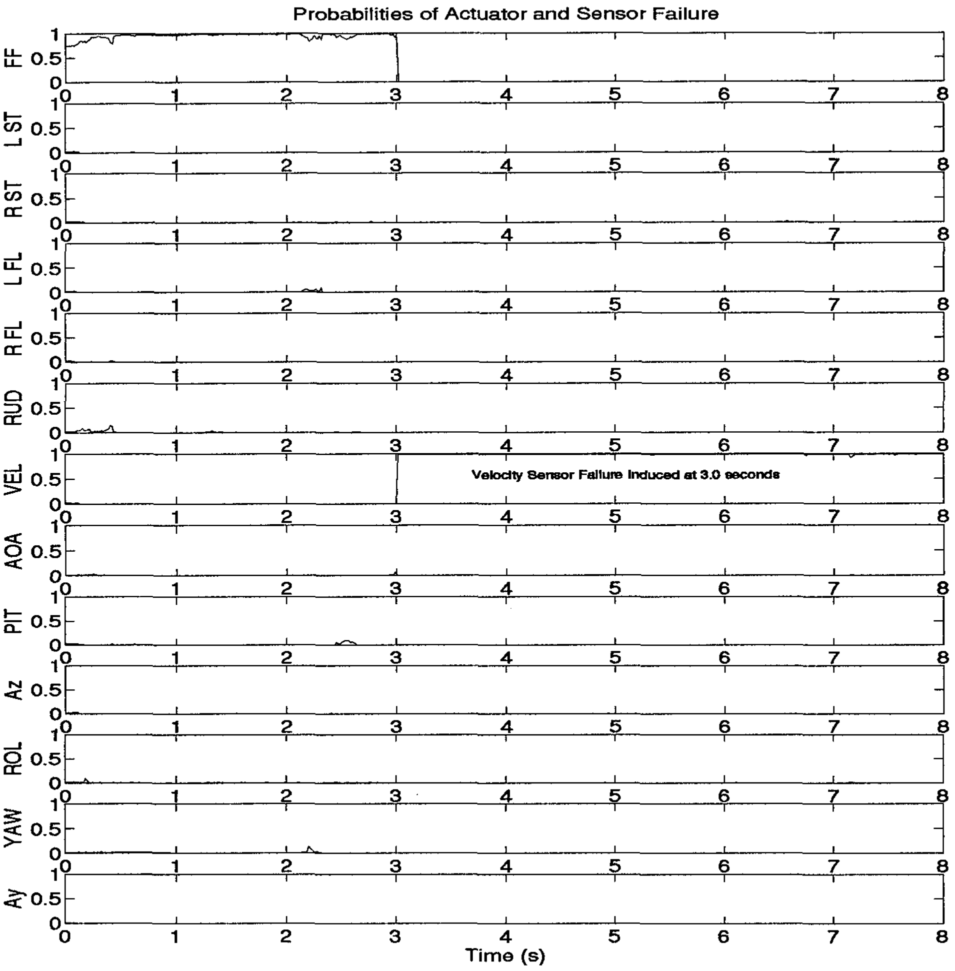


Figure 2. Probability Convergence: an Example Plot

  - *Are convergence times improved or degraded as a result of the higher order and nonlinear truth model?*
  - *Are there any heretofore unseen effects introduced as a result of the new truth model?*

- Dual Failures
  - *Do the elemental probabilities still converge to solutions?*
  - *Are they correct?*
  - *Are convergence times path-dependent?*

  - *Are convergence times improved or degraded as a result of the higher order and nonlinear truth model?*
  - *Are there any heretofore unseen effects introduced as a result of the new truth model?*
- Residual Monitoring
  - *Can residual monitoring be used as an additional voter for sensor failures?*
  - *Can residual monitoring be used as an additional voter for actuator failures?*
  - *Are there any heretofore unseen effects introduced as a result of the new truth model?*

1.6.2 Flight Envelope Research. Since the MMAE is based upon perturbation equations of motion, we would expect some performance degradation when the aircraft deviates from trim. The extent of such degradation in this environment has yet to be investigated. Assuming an area of adequate performance around every trim condition, a discrete number of MMAE algorithms which are scheduled throughout the flight envelope can be hypothesized. Figure 3 depicts this situation.

The flight envelope is covered by discrete MMAE's, in this figure four, each of which having an area of adequate coverage as shown. By filling up the envelope with such algorithms, entire coverage can be achieved. Some research questions in this area include:

- *How quickly does performance degrade off trim?*
- *Is the degradation flight-condition-dependent?*
- *Is a scheduled MMAE feasible based on the number of discrete trim conditions required?*
- *Can the scheduling mechanism be tied to the FCS gain scheduling being done at present?*

1.7 Scope

The scope of this thesis has been determined for the most part by time. As a result, the research has proceeded in steps. The first step is to rehost the MMAE algorithm into the SRF

Hypothetical F-16 Flight Envelope

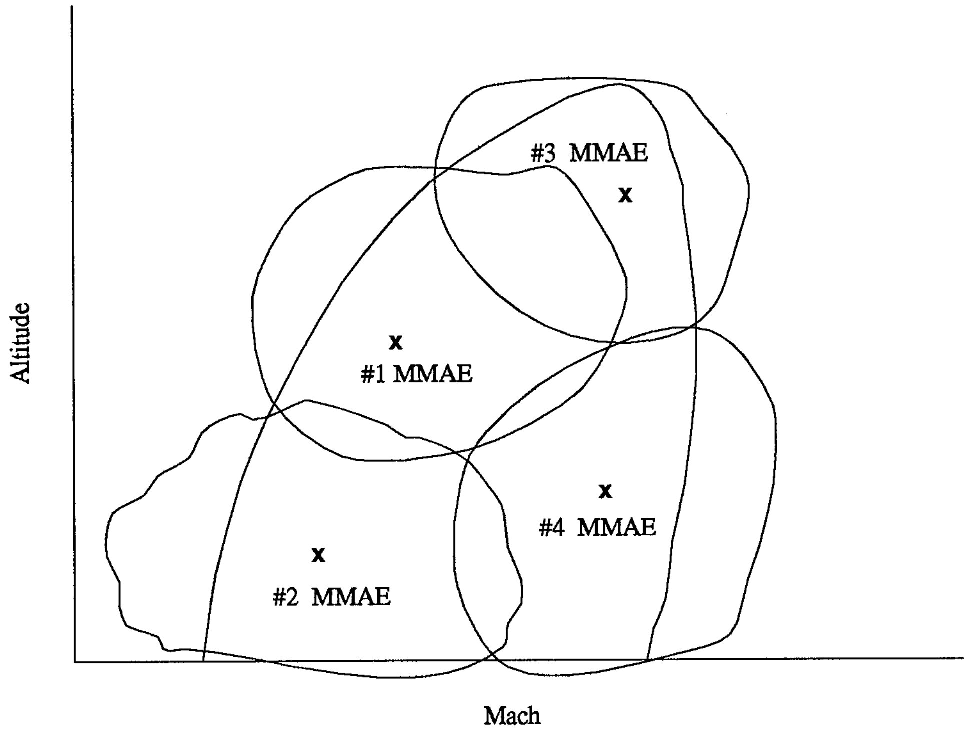


Figure 3. Exaggerated Discretization of Flight Envelope

environment and verify single failure operation. Dual failures are then added to the package in the second step and compared to the results of past research. Residual monitoring is then considered in Step 3 of this thesis. Upon successful implementation and completion of the failure performance demonstrations, research then begins on full envelope coverage in Step 4. In the event all of the above is concluded within the time allotted, time is to be spent again in Step 1, improving the newly created tool for future research efforts.

1.8 Limitations/Assumptions

By far, the most significant limitation/assumption lies in our acceptance of the SRF VISTA F-16 simulation. Since our reduced-order design models are in fact based on the real F-16, any flaws in the truth model representation will result in model mismatch and degrade algorithm performance. A rigorous approach would begin with a full validation of the truth model, but the simulation package is simply too large and complex, and time constraints are too restrictive for such a validation to be part of this research effort. In the following work, where not specifically stated, we have been forced to assume that the SRF VISTA F-16 is an accurate representation of an actual VISTA F-16. Other pertinent limitations/assumptions will be addressed as they arise within this document.

1.9 Report Format

This chapter's purpose has been to introduce the topic of research. Chapter 2 will provide a more in-depth presentation of the MMAE and give a short history of the algorithm's evolution. Chapter 3 develops the models used in this thesis. The SRF truth model is discussed in detail and the design models used within the elemental Kalman filters are fully developed. Chapter 4 presents the results. The data and plots are presented in chronological order, with narrative and observations included. Chapter 5 summarizes, draws conclusions, and discusses the results and provides recommendations for future research.

1.10 Chapter Summary

This chapter has introduced the subject and provided some of the direction in which this thesis research is headed. Motivation was given for the problem statement. Then a no-equations-allowed overview of the MMAE algorithm set the foundation for the objectives and research questions. The MMAE developed to date is sound, but really needs meaningful implementation. Demonstration of the MMAE's abilities in such an implementation is essential to ensure future Air Force interest. Administrative topics brought the chapter to a close. Scope and limitations/assumptions were pointed out, and a small section provided a road map for the rest of this document. Chapter 2 fills in the details omitted in this chapter's brief introduction.

2. Background and History

2.1 Chapter Overview

In this chapter we discuss the Multiple Model Adaptive Estimation (MMAE) algorithm in further detail. Specifically, the theory behind the Kalman filter and its application to this algorithm will be addressed in Section 2.2.1. Modifications to the algorithm are explained in Section 2.2.2. These performance enhancements are the techniques which have been tested and exploited in previous research efforts. The concept of dithering, which is artificially exciting the system state for better performance, is covered in Section 2.2.3. Following that, we shift into a bit of the history surrounding MMAE research. Emphasis focuses on the line of research performed at the Air Force Institute of Technology in Section 2.3.1. Other applications are discussed in Section 2.3.2. The chapter concludes with a summary.

2.2 Multiple Model Adaptive Estimation

Multiple Model Adaptive Estimation (MMAE) is based on the Kalman filter. A Kalman filter is capable of calculating a prediction of any system's behavior by knowing what has just happened to the system (sensor data), what the system is being asked to do (control data), and possessing a reasonable mathematical model of that system's behavior. Of course, its prediction is imperfect due to inherent uncertainty in reduced-order mathematical models, random behavior, and noisy sensor data. The Kalman filter attempts to minimize these uncertainties by including stochastic models within its calculations. Further details on the internal operations of a Kalman filter can be found in (20). A basic understanding of the Kalman filter is henceforth presumed, although the basic algorithm is presented in the next section.

Assume a generic system whose continuous, time-invariant dynamics and sampled-data measurement equations are given and defined below.

Continuous Dynamics Model:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{G}\mathbf{w}(t) \tag{1}$$

Sampled-Data Measurement Model:

$$\mathbf{z}(t_i) = \mathbf{H}\mathbf{x}(t_i) + \mathbf{v}(t_i) \tag{2}$$

Linearized equations of motion are contained in **A** and **B** matrices with **x** being the state vector and **u** the deterministic control input vector. **w** is a random vector of zero-mean, Gaussian, white noise inputs of strength **Q** (not shown). The noise inputs are brought into the state equations through the **G** matrix and represent uncertainty in the dynamics model. **z** is the vector of available measurements which are related to the state vector through **H** and corrupted by **v**. **v** is a vector of zero-mean, white, discrete-time, Gaussian noise of covariance **R** (not shown) which represents uncertainty in the measurement model. Noises **v** and **w** are assumed independent and thus uncorrelated. Typically, perfect knowledge of the entire state **x** is impossible and we must operate with knowledge of a reduced set of states which can be measured (with noise corruption) and other noise-corrupted measurements available which are combinations of state variables. Further discussion of the models, state vectors, and noises used in this thesis is postponed, but will be brought up again in full detail in Chapter 3. For now we can proceed to develop the algorithm assuming any generic system whose behavior can be described in the form of Equations (1) and (2).

2.2.1 Algorithm Development (Theoretical). Since this algorithm is to be applied in a digital environment, an equivalent discrete-time model (20) representation of the continuous, time-invariant dynamics and measurement equations is appropriate. Such a model produces discrete-time state and output values with characteristics that match those of Equations (1) and (2) at each discrete sample time. Note that the subscript k will be defined in the following paragraphs; for now let it be an index designating which model of K possible models ($k = 1, 2, \ldots, K$) is under consideration. (Again, a discussion of the generation of the thesis models is deferred until Chapter 3.)

Discrete Dynamics Model:

$$\mathbf{x}_k(t_{i+1}) = \mathbf{\Phi}_k \mathbf{x}_k(t_i) + \mathbf{B}_{\mathbf{d}k}\mathbf{u}(t_i) + \mathbf{G}_{\mathbf{d}k}\mathbf{w}_{\mathbf{d}k}(t_i) \tag{3}$$

Discrete Measurement Model:

$$\mathbf{z}(t_i) = \mathbf{H}_k(t_i)\mathbf{x}_k(t_i) + \mathbf{v}_k(t_i) \tag{4}$$

The state transition matrix is given by $\mathbf{\Phi}_k = e^{\mathbf{A}_k \Delta t}$, where Δt is the sample period. $\mathbf{B_d}$ and $\mathbf{G_d}$ correspond to discretized $\mathbf{B}$ and $\mathbf{G}$. The discrete dynamics model also contains $\mathbf{Q_d}$ (not shown) which corresponds to a discretized $\mathbf{Q}$.

Based upon system equations in this form and assuming the initial condition $\hat{\mathbf{x}}(t_0)$ is Gaussian with mean $\mathbf{x}_{k0}$ and covariance of $\mathbf{P}_{k0}$, the Kalman filter equations become:

Propagation Equations for State Estimate and Error Covariance:

$$\hat{\mathbf{x}}_k(t_{i+1}{}^-) = \mathbf{\Phi}_k(t_{i+1}, t_i)\hat{\mathbf{x}}_k(t_i{}^+) + \mathbf{B}_{\mathbf{d}k}(t_i)\mathbf{u}(t_i) \tag{5}$$

$$\mathbf{P}_k(t_{i+1}{}^-) = \mathbf{\Phi}_k(t_{i+1}, t_i)\mathbf{P}_k(t_i{}^+)\mathbf{\Phi}_k{}^T(t_{i+1}, t_i) + \mathbf{G}_{\mathbf{d}k}(t_i)\mathbf{Q}_{\mathbf{d}k}(t_i)\mathbf{G}_{\mathbf{d}k}{}^T(t_i) \tag{6}$$

Update Equations:

$$\mathbf{A}_k(t_i) = \mathbf{H}_k(t_i)\mathbf{P}_k(t_i{}^-)\mathbf{H}_k{}^T(t_i) + \mathbf{R}_k(t_i) \tag{7}$$

$$\mathbf{K}_k(t_i) = \mathbf{P}_k(t_i^-)\mathbf{H}_k{}^T(t_i)\mathbf{A}_k{}^{-1}(t_i) \tag{8}$$

$$\mathbf{r}_k(t_i) = \mathbf{z}_i - \mathbf{H}_k(t_i)\hat{\mathbf{x}}_k(t_i^-) \tag{9}$$

$$\hat{\mathbf{x}}_k(t_i^+) = \hat{\mathbf{x}}_k(t_i^-) + \mathbf{K}_k(t_i)\mathbf{r}_k(t_i) \tag{10}$$

$$\mathbf{P}_k(t_i^+) = \mathbf{P}_k(t_i^-) - \mathbf{K}_k(t_i)\mathbf{H}_k(t_i)\mathbf{P}_k(t_i^-) \tag{11}$$

where the $+$ and $-$ superscripts indicate just after and just before time sample t_i (t_{i+1}), respectively.

Having established a discrete-time Kalman filter of the form shown, there arise situations where the means of dealing with uncertainty, namely white noise inputs $\mathbf{w}$ and $\mathbf{v}$, is insufficient (21). In flight control for example, not only is there uncertainty due to noisy sensors and reduced-order modelling, but there can also be uncertainty in the status of the airframe itself. The filter's mathematical model is based on the assumption (hypothesis) of working actuators and sensors, but suppose they are inoperative. The Kalman filter model would assume an incorrect representation of the actual aircraft configuration and its predictions of behavior would become quite specious. This example illustrates the need for adaptive techniques in general and MMAE in specific.

Let $\mathbf{a}$ denote a vector of uncertain parameters in a given linear stochastic state model for a dynamic system. In the example above, the values of $\mathbf{a}$ represent all possible failure modes of the aircraft. The number of possible values that this vector assumes can become unmanageably large, even infinite in continuous applications, so effort must be taken to discretize the parameter space. In this thesis, the parameter space is limited to the fully functional aircraft, five actuator failures, seven sensor failures, and combinations thereof in an F-16 flight control system. Failures are considered to be full or "hard" failures; partial or "soft" failures can be handled through a weighted average of outputs from the associated "hard" failures and fully functional models, as discussed somewhat in Section 1.4. By defining a hypothesis conditional probability $p_k(t_i)$ as the probability that $\mathbf{a}$ assumes the value $\mathbf{a}_k$ (for k = 1,2...,K: discrete possible values), conditioned on the observed measurement history to time t_i:

$$p_k(t_i) = Prob(\mathbf{a} = \mathbf{a}_k | \mathcal{Z}(t_i) = \mathcal{Z}_i) \tag{12}$$

where $\mathcal{Z}(t_i)$ is the history of all measurements from $\mathbf{z}(t_1)$ to $\mathbf{z}(t_i)$. It can be shown that $p_k(t_i)$ can be evaluated recursively for all k via the iteration:

$$p_k(t_i) = \frac{f_{z(t_i)|\mathbf{a},\mathcal{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_k, \mathcal{Z}_{i-1})p_k(t_{i-1})}{\sum_{j=1}^{K} f_{z(t_i)|\mathbf{a},\mathcal{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_j, \mathcal{Z}_{i-1})p_j(t_{i-1})} \tag{13}$$

The above equation tells us that the probability is based on its previous probability, weighted by the density function which describes a predicted $\mathbf{z}$ given $\mathbf{a}_k$ and $\mathcal{Z}$. The measurement history random vector $\mathcal{Z}(t_i)$ is made up of partitions $(\mathbf{z}(t_1), \ldots, \mathbf{z}(t_i))$ which correspond to the measurement vectors received by the filter at sample times $t_1, \ldots, t_i$. The denominator is simply the sum of all such values corresponding to each point in the parameter space and is used to normalize the resulting probabilities. As a result, we can expect the probabilities to behave in the following manner:

$$p_k(t_i) \geq 0 \tag{14}$$

$$\sum_{k=1}^{K} p_k(t_i) = 1 \tag{15}$$

At this point one should make clear that, in this thesis, the parameter space denoted by $\mathbf{a}$ never takes on any *numerical* values. When we say $Prob(\mathbf{a} = \mathbf{a}_k|\mathcal{Z}(t_i) = \mathcal{Z}_i)$, we mean in words: the probability that the correct hypothesized actuator or sensor failure status is ...(such and such) based on the history of sensor measurements taken.

Now assuming that at time t_i, the $p_k(t_i)$ have been calculated, the Bayesian minimum mean square error estimate of the state vector is a probabilistically weighted average given by:

$$\hat{\mathbf{x}}(t_i^+) = E[\mathbf{x}(t_i)|\mathcal{Z}(t_i) = \mathcal{Z}_i] = \sum_{k=1}^{K} \hat{\mathbf{x}}_k(t_i^+) p_k(t_i) \tag{16}$$

where $\hat{\mathbf{x}}_k(t_i^+)$ is the state estimate generated by Kalman filter k based on the assumption that the parameter value corresponds to $\mathbf{a}_k$. The process described to this point lends itself quite nicely to the architecture displayed in Chapter 1's Figure 1. Figure 4 further illustrates the MMAE algorithm and its association with this research. All elemental Kalman filters associated with the single failure case are shown.

The Multiple Model Adaptive Estimator algorithm is a bank of Kalman filters operating in parallel, each based on a different hypothesized point in a parameter space ($\mathbf{a}_k$). All are given the same information, namely the last vector of sensor values $\mathbf{z}_i$, and the current vector of system input values $\mathbf{u}$. (Actually, they have had equal access to the entire measurement time history $\mathcal{Z}_i$ as well.) The key to the algorithm's operation lies in the filter-computed residuals and Equation (13). Each filter uses the input information and its model of system behavior to produce a prediction of the system state one time step in the future. This prediction is transformed into a prediction of sensor values and compared to the actual measured sensor values, as shown in Equation (9). Figure 4 shows that, in addition to the estimates of state $\hat{\mathbf{x}}_k(t_i)$, each filter outputs the computed residual vector $\mathbf{r}_k(t_i)$ for probability evaluation.

Looking once again at Equation (13), we see that the numerator density (shown below) is actually the statistical description of each filter's residuals:

$$f_{\mathbf{z}(t_i)|\mathbf{a},\mathcal{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_k, \mathcal{Z}_{i-1}) = \frac{1}{(2\pi)^{\frac{m}{2}} |\mathbf{A}_k(t_i)|^{\frac{1}{2}}} e^{[-\frac{1}{2}\mathbf{r}_k(t_i)^T \mathbf{A}_k{}^{-1}(t_i)\mathbf{r}_k(t_i)]} \tag{17}$$

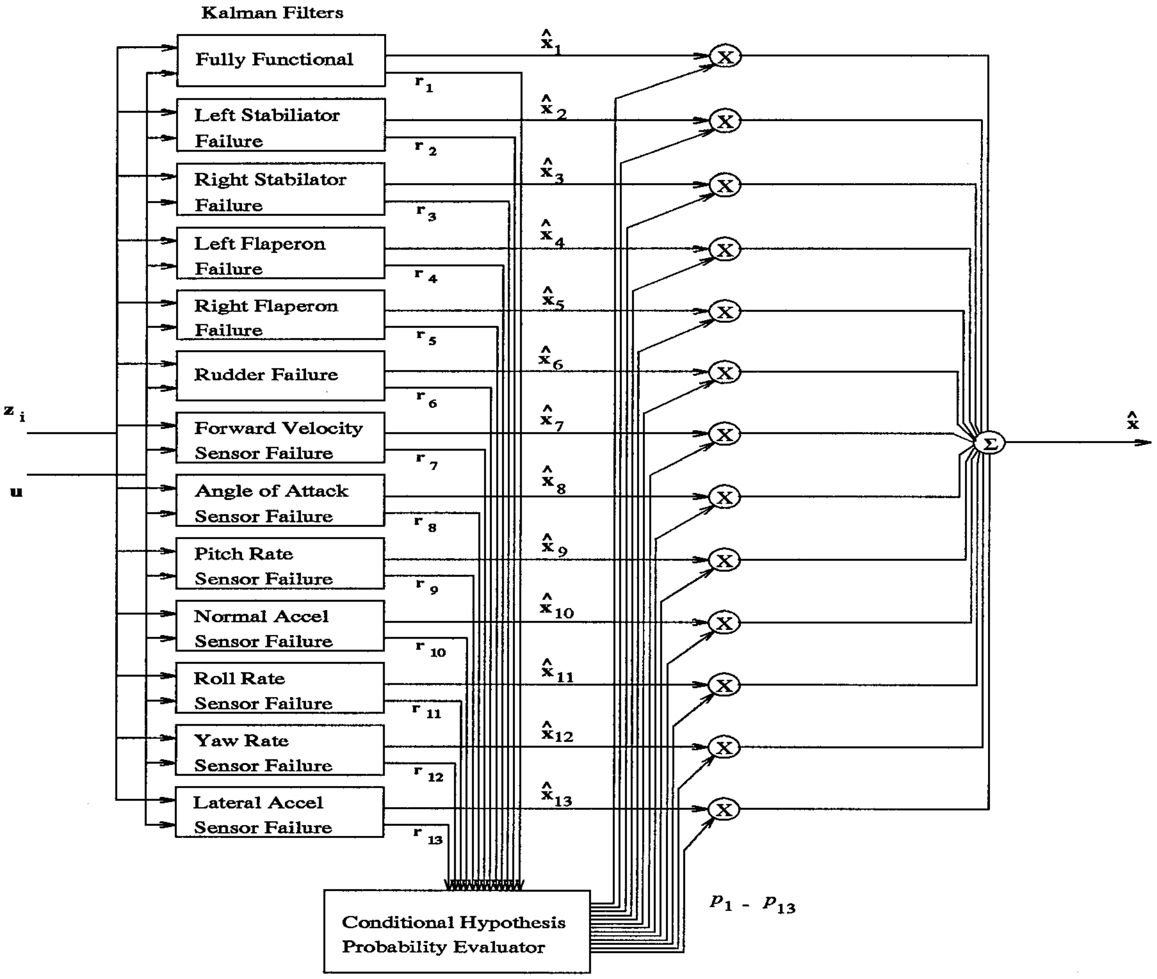


Figure 4. Thesis application of the MMAE algorithm

where $\mathbf{A}_k$ is the covariance of the residuals as computed by the k-th Kalman filter, as in Equation (7). In operation, one expects the residuals of the filter which best matches the true system status to have a mean squared value most in consonance with its internally computed covariance, $\mathbf{A}_k(t_i)$:

$$L_k(t_i) = \mathbf{r}_k(t_i)^T \mathbf{A}_k{}^{-1}(t_i)\mathbf{r}_k(t_i) \approx m \tag{18}$$

For those filters which are mismatched, the likelihood ratio L_k gets larger than m, where m is the dimensionality of $\mathbf{A}_k$, resulting in a smaller value of $f_{\mathbf{z}(t_i)|\mathbf{a},\mathcal{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_k, \mathcal{Z}_{i-1})$ and corresponding reduction in $p_k(t_i)$. As shown in Figure 4, the probabilities calculated using Equation (13) are then used to weight the individual state estimates using Equation (16), providing the best estimate of system state.

2.2.2 Algorithm Development (Experimental). The theoretical development shown in part in the previous section doesn't tell the entire story. There are a handful of algorithm modifications, or *performance enhancements*, which have been developed during the course of past research. As the moniker suggests, their incorporation has been demonstrated beneficial in improving the algorithm's ability to distinguish between failures and also to do so in a timely manner. Of course, any results from a modified algorithm lose theoretical statistical optimality, but if the goal is improved algorithm performance, this is an acceptable tradeoff. The following sections discuss the performance enhancements most likely to be utilized in this thesis: Modified Bayesian Form, Beta Dominance, Hierarchical Modeling, Increased Scalar Penalties, Probability Smoothing, and Kalman Filter Retuning.

2.2.2.1 Modified Bayesian Form. The final probability weighted average of the state estimate computed using Equation (16) has been found to require modification due to practical considerations. First, in looking at Equation (13) again, we see that due to the recursive nature of the calculation, steps must be taken to prevent any of the p_k's from getting too small. A filter whose probability is very small will require many sample periods of iteration to grow in the event its failure condition arises in the real world. That translates into unacceptably slow algorithm performance. Worse yet, a p_k which ever becomes zero by this equation computation will stay at zero and become forever locked out. The algorithm would lose sensitivity to that failure. What is needed then, is a lower bound on the p_k's. In fact, this has been done successfully and the value currently in use is 0.001 (9, 14, 24, 26–28, 30, 32).

When looking at Equation (17), a second consideration is the effect of the contributions of incorrect filters. Especially now that an artificial lower bound exists for each filter regardless of correctness, we must consider a means of excluding filter contributions from filters whose estimates are known to be incorrect. This can be accomplished in practice by including a second lower bound slightly higher than 0.001. Past research has found 0.003 to be acceptable (24,30). The effect of the second lower bound is to include only those filters whose probabilities exceed 0.003 in the blended estimate. In most cases, this will only be one filter, or perhaps two if the failure isn't hard over.

The combination of these bounds demonstrates the transition from theoretical Bayesian form to the modified Bayesian form. Again, all theoretical optimality is lost, but significant performance enhancement is gained. The modified Bayesian form will be incorporated into this thesis.

2.2.2.2 β Dominance. An interesting phenomenon occurred during past research which led to this performance enhancement (1, 24, 30). In the event that all of a bank's filters' likelihood ratios were approximately the same (Equation (18)), then the leading term, called the "β" term, of Equation (17) began to influence the algorithm's decisions directly, or β dominance was said to occur. The effect was a predisposition towards declaring sensor failures. This phenomenon is easy to understand because of the fact that sensor failure $\mathbf{A}_k$ matrices are smaller than those corresponding to actuator failures or no-failure conditions. (Remember: sensor $\mathbf{A}_k$'s are generated from $\mathbf{H}_k$ which have a row of zeros for the case of sensor failures.) Since Equation (13) is normalized by the sum of the numerator terms of all of the filters, it was hypothesized and demonstrated that the leading term, $\frac{1}{(2\pi)^{\frac{m}{2}}|\mathbf{A}_k(t_i)|^{\frac{1}{2}}}$, could be stripped from the calculations entirely. Removing this β term eliminated the predisposition towards sensor failures, and this modification will be included in this thesis.

2.2.2.3 Hierarchical Modeling. As was discussed earlier, one subject of ongoing concern in multiple model adaptive estimation is parameter space discretization. Too coarse a discretization can result in poor performance, while too fine a discretization can result in an unmanageably large computational burden. How then do we handle a parameter space that includes two-failure scenarios as well as one-failure hypothesis? Certainly it is possible for two actuator or sensor failures to exist at any time. In order to have a bank of filters which can handle both single and double failures then, one would need $1 + K + \frac{K!}{(K-2)!2!}$ filters on line. K is the number of hypothesized single failures chosen from the parameter space. That quickly becomes too many for practical operation. In this application, with K = 12, the result is 80 filters on line: clearly unacceptable.

A couple of different solutions to this problem exist. The one which is best suited to this application and has worked well in past research is hierarchical modeling (14, 26, 27, 32, 33). Hierarchical modeling uses $K+1$ banks of $K+1$ filters each, with only one bank ever on line at a time. One bank, Level 0, assumes no failure initially and looks for a single failure. K other banks all assume one of the single failures up front and look for the second failure. As an example, consider a fully functional aircraft. The filter bank corresponding to that no-failure condition and K single failures is on line. If the algorithm declares a failure in a filter, a new bank of filters is brought on line. This bank looks for the second failure assuming the first failure condition in all filters except one. The hypothesis of no failure is always included as one filter in the banks to allow for a reversal in failure declaration, a "back door" out of the hierarchy to the previous level. Figure 5 shows the hierarchical structure. If the probability associated with failure condition $\mathbf{a}_1$ exceeds an upper threshold for a prespecified number of sample periods (in our research, one), a failure is declared and the appropriate bank which has been designed for the isolated single failure $\mathbf{a}_1$ and any other possible second failure is called on line.

Hierarchical modeling is an effective way of addressing the parameter space discretization problem. Having only $K+1$ filters on line at any time greatly reduces the computational burden, and with memory becoming cheaper, filter storage isn't a problem. This performance enhancement will also be used in this thesis.

2.2.2.4 Increased Scalar Penalty. Another performance enhancement which warrants consideration increases the magnitude of the scalar coefficient which precedes the likelihood function in Equation (17) (9, 23). Theoretically this value is $-\frac{1}{2}$, but the effect of increasing the

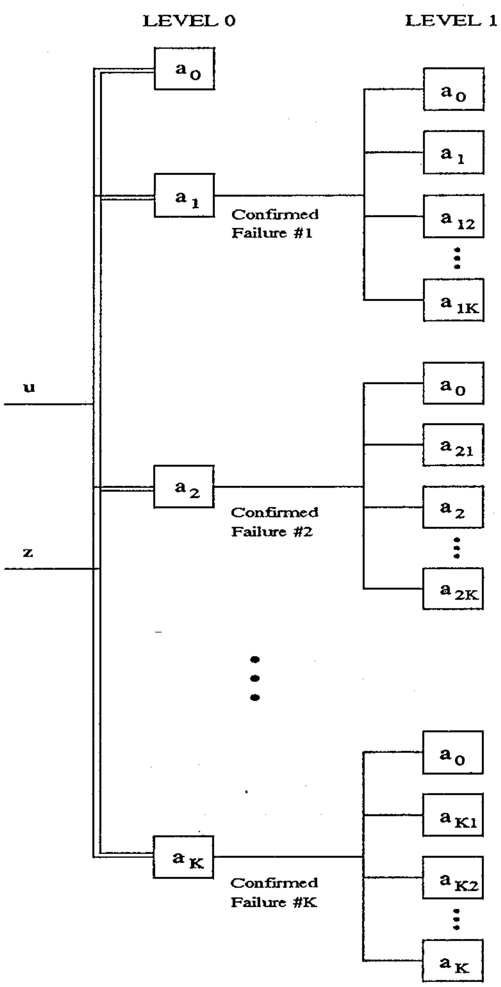


Figure 5. Hierarchical Structure

scalar is to accentuate or exaggerate any trends in the scalar $\mathbf{r}_k(t_i)^T \mathbf{A}_k{}^{-1}(t_i)\mathbf{r}_k(t_i)$, and thus to increase sensitivity to the onset of failures. In cases of slowly converging failure declarations, this can quite markedly increase the speed with which the algorithm makes its decisions. The downside is that it also accentuates abberations. This can result in spiking and false alarm declarations. Previous research has found a value of -1 works best. This thesis will consider both $-\frac{1}{2}$ and -1, depending on the algorithm's speed of response.

2.2.2.5 Probability Smoothing. Somewhat related, but at the other end of the spectrum from increased scalar penalty is probability smoothing. Where increasing the scalar penalty speeds sluggish response but can lead to false alarms, probability smoothing minimizes false alarms but can lead to sluggish performance. The technique creates a moving window of probabilities which is averaged to arrive at a smoothed overall probability. The effect is to eliminate any jumpy behavior which could result from single bad measurements. The window size used with good results in past research was 10. This thesis will consider using probability smoothing if false alarms become a problem. A distinction should be made between using averaged p_k's in the blending Equation (16) and using averaged p_k's off-line for failure declaration and bank switching only. Previous research efforts have performed the smoothing and then made failure declarations outside the MMAE algorithm after the state estimate has already been generated; unmodified p_k's are retained in Equation (16)'s computations (9,14). This thesis will continue that strategy.

2.2.2.6 Kalman Filter Re-tuning. A final performance enhancement to be considered deals with filter tuning. Previous research has shown that occasionally, a particular failure causes momentary bouncing of the p_k values. The algorithm is confused at the onset of failure, but eventually smoothes out nicely. The bouncing is undesirable, especially if the hierarchical modeling is being used because banks are being switched in and out as the probabilities go up and down rapidly. That is the purpose of not switching banks until the probability exceeds our threshold for a prespecified number of sample periods as discussed in Section 2.2.2.3. However, if the bouncing is such that the incorrect indication lasts for more than a few sample periods (10), further increasing the size of this "window" will unnecessarily slow performance for other failures. To handle this problem, the filters can be de-tuned in the uncertainty corresponding directly to the troublesome failure. The filter internal noise variances are increased to, in effect, desensitize it. (Such detuning could also be combined with a "window" of size greater than one as the designer sees fit.) Results show that bouncing can be reduced at a cost of slightly slower performance. If this thesis encounters such failures, Kalman filter de-tuning will be considered.

2.2.3 Dithering. A dither signal is an intentional stimulus to the aircraft actuators that has as its purpose to improve the algorithm's failure identification capability. Working the controls excites the system state and provides a means of differentiating between the various failure modes. This is especially important when the failure is an actuator. For example, if there is no actuation of the rudder, like during steady level flight, it would be impossible to know that a failure existed. Constantly working the controls then gives the filters something for which to look.

Past research has focused for the most part on determining the most effective continuous dither signal. Sine waves, modified sine waves, square waves, triangle waves, and modified pulse trains were researched with sine waves showing the best effectiveness (26,28). Also of concern was the magnitude of dither as well as the dither philosophy. Two scenarios have been hypothesized. One is a continuous, automated, subliminal dither. Such a signal would provide around-the-clock failure detection and isolation with no pilot intervention required and without any pilot discomfort. Of course, the limiting factor is the magnitude of the dither. If actuation is too strong, or at certain frequencies, the pilot would experience an uncomfortable oscillatory ride. Past research has defined subliminal to be $\pm.1$ g's in the longitudinal channel and $\pm.2$ g's in the lateral channel (26,28). On the other hand, if the actuation is too small, algorithm performance may suffer. The goal is to find the middle ground or another alternative. The second scenario is an emergency shake up. At the pilot's initiation, following engagement perhaps, a non-subliminal dither signal of specified duration would more violently shake up the system to identify any failures positively. This second scenario is easier to implement than the first. With enough system excitation, any failure could be identified.

Another scenario which concerns appropriate dither selection is the case of failed actuators. When the first failure happens to be an actuator, the effectiveness of a chosen dither scheme is diminished by the corresponding loss of control authority. In some cases, the resulting drop in system excitation has made a second failure difficult and sometimes impossible to detect (26). The proposed solution is to adjust, or schedule, the chosen dither scheme if the first failure is an actuator such that the level of system excitation remains high enough for further failure identification. This interesting topic will not be further pursued in this thesis.

2.2.4 Residual Monitoring. Residual monitoring is a technique which can increase algorithm effectiveness by providing an additional failure declaration vote. Kalman filtering theory predicts that residuals in well-matched filters will be zero-mean, white, Gaussian and of covariance $\mathbf{A}$, where $\mathbf{A}$ is computed via Equation (7). It has been demonstrated that, in fact, this behavior is exhibited in filter residuals which correspond to the correct failure hypothesis (26). Specifically, Menke showed that, up until the time a failure was introduced, the fully functional hypothesis filter residuals were zero-mean, white, Gaussian and of covariance $\mathbf{A}$ as computed within that filter. Upon failure induction, the fully functional filter's residuals ceased such behavior, and concurrently, the residuals in the filter corresponding to the correct failure hypothesis began to exhibit the behavior.

Menke further proposed some simple tests which could be used in obtaining the corroborative vote: a "σ-bound" test, a "zero-meanness" test, and a "whiteness" test (26). The "σ-bound" test is based on the fact that, theoretically, well-matched filter residuals fall within their 2σ bound 95% of the time. Over a given period of time, scalar residual samples are checked against the 2σ threshold. If 95% are within the bounds, a failure vote can be issued. In the "zero-meanness" test, the same samples can be temporally averaged to see whether or not a bias exists. And similarly, the "whiteness" test can use the same values, with any mean subtracted off, and count the number of zero crossings (sign changes). The higher the count, the more white a residual appears to be. These simple tests form the basis for providing an additional vote.

Sensor failures show their effects directly in the appropriate scalar residual, while an actuator failure's effects propagate through the system and appear indirectly in all residuals. Thus, we would expect this technique to work better for sensors than actuators. Menke showed, however, that by looking at certain residuals in well-matched filters, the technique also provided the additional vote for actuator failures as well.

Another phenomenon which Menke demonstrated in scalar residuals was the appearance of a sinusoidal oscillation within some mismatched filter residuals (26). The frequency of oscillation was the same as that of the input dither sinusoids. The dither's effect on airplane motion is in turn oscillatory, therefore the measurement vector entries will also show oscillation at that frequency. In mismatched filters, the model's prediction of $\mathbf{H}_k(t_i)\hat{\mathbf{x}}_k(t_i^-)$ (see Equation (9)) doesn't correctly match $\mathbf{z}_i$, so some of the oscillation gets through. This is because the $\mathbf{B}_{\mathbf{d}k}\mathbf{u}(t_i)$ term used to compute $\hat{\mathbf{x}}_k(t_{i+1}^-)$ in Equation (5) are based on an erroneous failure hypothesis. If necessary, this phenomenon could also be exploited in residual monitoring schemes. Using the data from the simple tests just discussed, one can seek the appearance or nonappearance of this dither oscillation. Knowing the frequency ahead of time, this is quite easy. This research effort will observe the filters' scalar residuals to see if these techniques remain viable as potential additional voters.

2.3 MMAE Research

In 1976, A.S. Willsky surveyed a broad spectrum of failure detection and isolation schemes (35). One of his conclusions predicted that multiple model techniques had the potential to yield the best performance over the widest class of failures. With the advent of low cost digital computing and parallel processing, the time has arrived to bear this prediction out. In fact, the U.S. Air Force has taken an interest in investigating multiple model techniques applied to flight control. A line of fruitful research has taken place at the Air Force Institute of Technology (AFIT).

2.3.1 Flight Control at the Air Force Institute of Technology. In 1988, longitudinal control of the Short Take-off and Landing (STOL) F-15 using a MMAC was investigated by Pogoda and Stevens (24, 25, 30, 32). Pogoda used Command Generator Tracking, Proportional plus Integral, Linear Quadratic (CGT/PI/LQ) elemental controllers to maintain stable control in spite of single failures introduced during the landing phase of flight. Failures included stabilators, "pseudo-surfaces" (which are a combination of canards, ailerons, and flaps), a pitch rate sensor, and the no-failure condition. Stevens continued the research by expanding the list of failures to include reverser vanes, a velocity sensor, and a flight path sensor. He also investigated soft, or degraded mode, failures and multiple failures. The hierarchical structure was introduced to handle dual failures. Both researchers concluded that the MMAC was capable of detecting and isolating failures, and maintaining stable control after reconfiguring the FCS based on the failure identified. Stevens also noted that the multiple model structure indeed blended outputs appropriately when the failures were soft, and the hierarchical structure was able to switch banks quickly enough that stability was maintained.

Continuing these pursuits, Stratton and Menke developed a MMAE for use with the Variable In-Flight Stability Augmentation (VISTA) F-16 (26–28, 33). This aircraft model differed significantly from the STOL F-15 in that both the longitudinal and lateral axes were being simulated and the F-16 already contained a FCS. One of their goals then, was to use MMAE-based control with an existing controller and focus on the ability of the MMAE to identify failures quickly and accurately, and also provide a good estimate of state to the FCS. Stratton used a flight condition of .8 Mach and 10,000 ft altitude and Menke moved to .4 Mach and 20,000 ft. This represented a reduction in dynamic pressure, thus creating more of a challenge for the algorithm. Their research looked at single and dual, hard and soft failures in actuators and sensors. Using 13 elemental filters in all, the failure modes were fully functional aircraft, left and right stabiliator and flaperon and rudder actuators, and forward velocity, angle of attack, pitch rate, normal acceleration, roll rate, yaw rate, and lateral acceleration sensors. Also, their studies investigated the use of dithering to excite system state and thus enhance failure identification during benign portions of flight as well as algorithm performance during normal aircraft maneuvering. Their results indicated that with enough state excitation, all single failure modes and most dual failure modes were detected and isolated with no ambiguity. The dithering may then be scheduled only when needed.

In 1992, the algorithm was moved toward the LAMBDA unmanned research vehicle (9,23). This is a model airplane developed by the Flight Controls Division of the Flight Dynamics Directorate, Wright Laboratory, as an affordable, flexible research vehicle for testing and demonstrating flight control concepts, devices and systems (9). Hanlon successfully applied the MMAE to a computer model of the LAMBDA URV for the single failure mode only (9,23). The failures simulated were right and left elevator, aileron, and rudder actuators, and velocity, angle of attack, pitch rate, pitch angle, sideslip, roll rate, roll angle, and yaw rate sensors. Lane expanded the algorithm application to include dual failures using the hierarchical structure and streamlined it for future flight testing on the actual URV with a Quantitative Feedback Theory (QFT) designed controller installed (14).

2.3.2 Other and Elsewhere. In addition to flight control at AFIT, multiple model techniques are also being applied to target tracking (10), control of flexible space structures (5,7), robotics control (15), and for use in head mounted displays (31). Other non-military applications have included autonomous monitoring of electrocardiograms (37), failure detection in a pressurized water fusion reactor (4), detection of incidents on freeways (36), marine exploration (8), and air traffic control (16).

2.4 Multiple Model Adaptive Control

Similar yet different from the Multiple Model Adaptive Estimation (MMAE) algorithm is the Multiple Model Adaptive Control (MMAC) algorithm. As the names suggest, the goal of the former is to provide a state estimate and the latter, to provide a control law. Figure 6 shows the MMAC structure. It is nearly identical to the MMAE structure except that inserted in each filter's output path is a controller designed specifically for that failure status. Instead of blending state estimates at the output then, the individual control laws are optimally blended for best performance. Individual control blocks may be designed by any means available: QFT, LQG, state-space, etc.

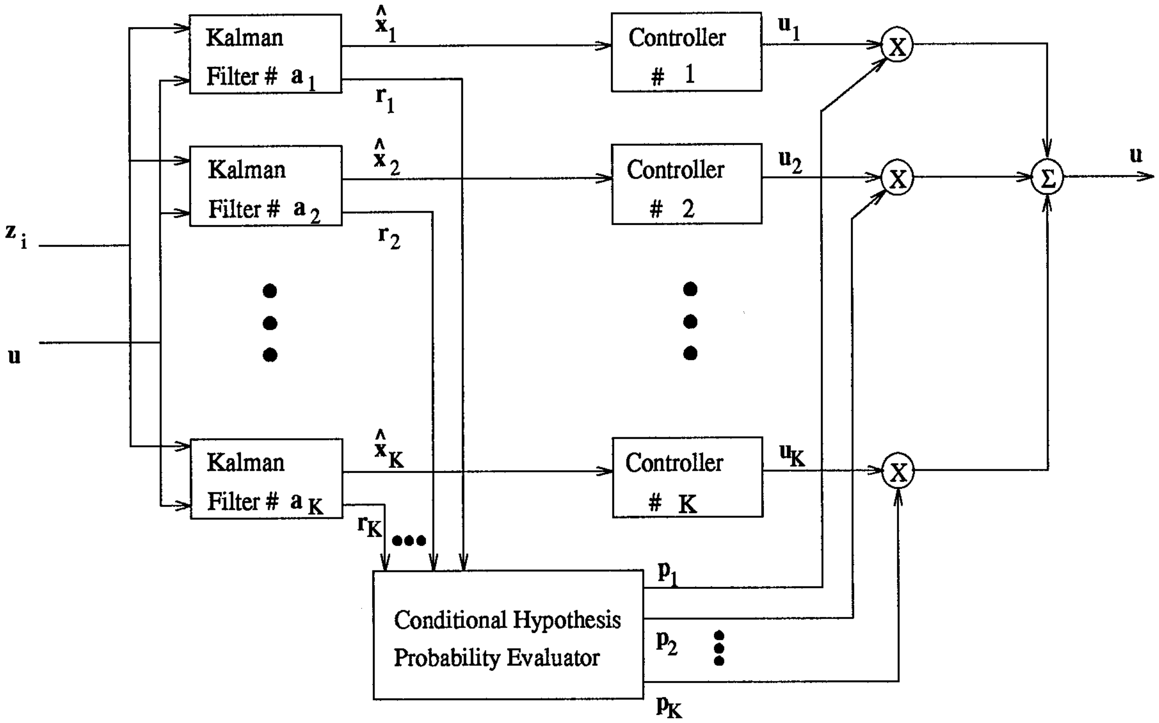


Figure 6. Multiple Model Adaptive Controller

Research has shown promise using command generator tracking, proportional plus integral, linear quadratic Gaussian, and CGT/PI/LQG controllers.

All other considerations discussed in the preceding sections apply equally to the MMAC as the MMAE. This thesis will not employ the MMAC algorithm. Instead, the MMAE algorithm will be used in MMAE-based control. MMAE-based control features a single separate controller being fed by the output of the MMAE. In this case, the controller is the Block-40 flight control system of the F-16. Figure 7 depicts MMAE-based control.

2.5 Chapter Summary

The preceding chapter has developed in part, the MMAE algorithm used in this thesis. In addition to the theory, practical experience has shown that modifications to the theoretical algorithm greatly enhance system performance. The performance enhancements likely to be utilized in this thesis were therefore discussed. A short history of MMAE research was given, focusing on

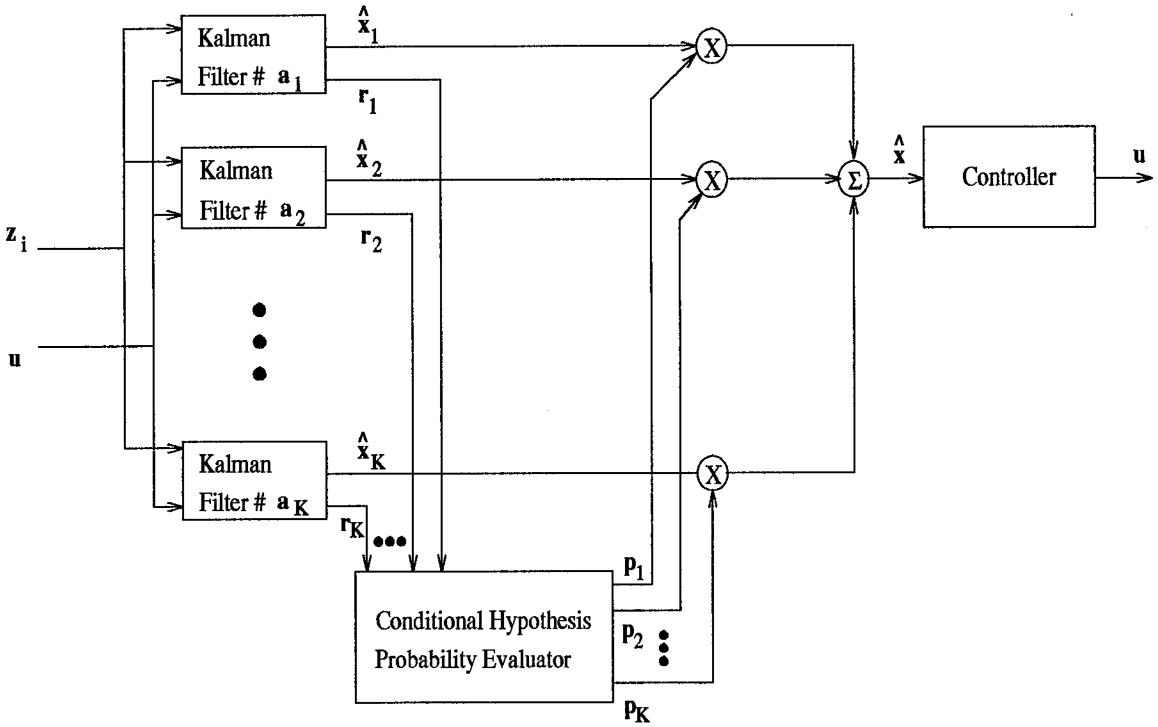


Figure 7. MMAE-based Controller

the line taken at the Air Force Institute of Technology. The MMAC was then discussed briefly, but will not be utilized in this thesis. Model development, to be taken up in the next chapter, will instead use the MMAE-based control algorithm of Figure 7.

3. Methodology and Model Development

3.1 Chapter Overview

For this thesis, there are two top level models with which we are most concerned. One is the truth model, which is the best representation available of a VISTA F-16, its flight control system and atmospheric disturbances. The other is the design model, which is a reduced order representation of the truth model. The design model is used in the MMAE Kalman filters. This chapter first describes the truth model in Section 3.2. Then, in Section 3.3, the reduced order model is developed in detail. Finally, the chapter concludes with a summary.

3.2 Truth Model - SRF

The truth model used in this thesis is the F-16 VISTA simulation, which is installed in the Simulation/Rapid-Prototyping Facility (SRF) at the Flight Control Division of the Flight Dynamics Directorate, Wright Laboratory. A truth model is the most complete and accurate simulation model available for a given physical process. As the name suggests, it serves as reality within a simulation environment. The SRF VISTA F-16 is a near-real-time, six-degree-of-freedom, nonlinear aircraft simulation. It is composed of General Dynamics' F-16 VISTA simulation software, Calspan's Variable Stability System model software (unused in this thesis), the SRF simulation template, and standard simulation software developed by and for the Flight Dynamics Directorate. Pertinent to this thesis are the equations of motion, the FCS simulation, and the models used for atmospheric disturbances. The following sections describe this code as well as the interfaces between the SRF and MMAE software, and the code used to insert failures into the SRF environment.

3.2.1 Equations of Motion. As mentioned in Chapter 1, Section 1.8, to a large extent, the SRF VISTA F-16 is a "black box" by necessity. No more is this the case than in the equations of motion. Unlike past research which used linearized truth models, equations of motion in this simulation are not as simple as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

This simulation calculates stability derivatives and moment coefficients through the use of interpolation and look-up tables. These are then utilized in fully nonlinear dynamic motion equations too numerous to detail. Further, inclusion of nonlinear effects such as atmosphere, operation of the engine, and a fourth order integrated servo-actuator (ISA) exemplify the level of complexity which leads to acceptance of the equations of motion in a black box manner. Further, our future modelling is based on a set of perturbation equations generated through a linearization routine. Full understanding of the equations of motion isn't required at this point. One exception however is the actuator models, which are a critical piece of our reduced order design model and aren't provided by the linearization routine. Section 3.3.1 explains our approach to modelling the ISA's. The next two sections discuss the Flight Control System and disturbance models with which we are more familiar.

3.2.2 Flight Control System. The Flight Control System (FCS) used in the SRF VISTA F-16 simulation is the Block 40, digital, fly-by-wire system. For proprietary reasons, actual drawings and specifications are intentionally omitted from this discussion. A higher level functional block diagram is displayed in Figure 8, and the general features are discussed in the following paragraphs.

The control system is broken down into functional areas which include pitch, roll, and yaw control, leading and trailing edge flaps, and command mixing. (Gun compensation is not included in the simulation.) Seven critical sensed inputs are used by the FCS in computing the actuator commands. The sensors are, once again, forward velocity, angle of attack, pitch rate, normal acceleration, roll rate, yaw rate, and lateral acceleration. Forward velocity is not shown explicitly in Figure 8 because its use is in computing the FCS gains which also aren't shown. The MMAE used in this thesis detects and isolates failures in all seven of these sensors. Also shown in Figure 8 are the six actuators modeled within the simulation. These are the left and right stabilators,

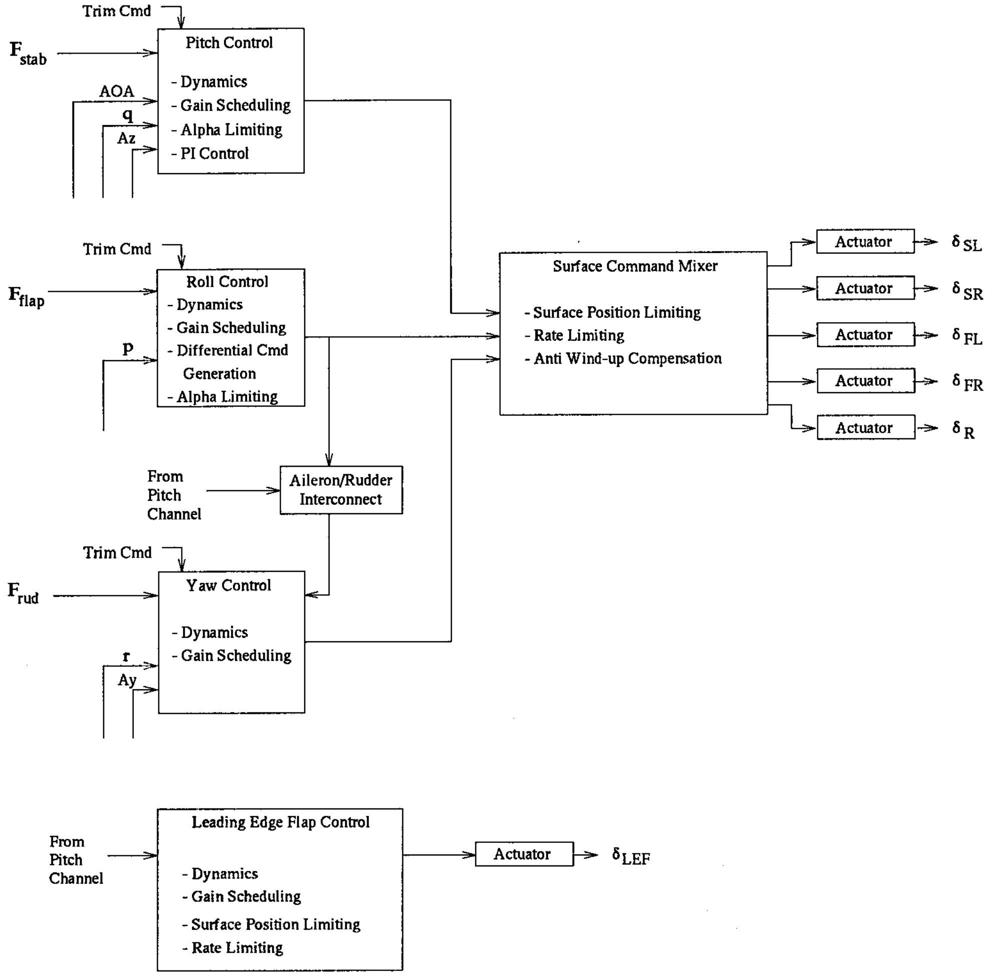


Figure 8. VISTA F-16 Functional Block Diagram

left and right flaperons, rudder, and leading edge flap. Failures in all but the leading edge flap are included in this thesis.

In all, thirteen inputs are used by the FCS; the seven sensor inputs, three pilot commands, the three forces F_{stab}, F_{flap}, and F_{rud}, and three trim commands. As shown in Figure 8, there is a pilot command and trim command in each control channel. General processing of these inputs within each channel include command limiting, gain scheduling, filtering, coupling, mixing and anti-wind-up.

In the pitch control channel, the pilot's stick force input is converted by a command gradient and summed with the trim command. These are then limited and filtered before being summed with sensor-generated feedback signals. These feedbacks are the product of angle of attack, pitch rate, and normal load factor, and also include alpha (angle of attack) limiting based on roll rate and flight condition. The sum is then sent through a proportional plus integral (PI) elevator command generator, which includes more limiting before being sent to the surface command mixer.

In the roll control channel, pilot stick force is also converted by a command gradient, but limited and filtered prior to summation with trim and sensor-generated feedback signals. Command limiting is based on pitch commands, and sensor feedback is based on roll rate. The sum is again limited before being sent to the surface command mixer. The same signal is also used to generate the differential tail command, and Aileron Rudder Interconnect (ARI).

In the yaw control channel, pedal force input is converted by a command gradient and filtered before being summed with the trim command and sensor-generated feedbacks. Sensor feedbacks in the yaw channel include yaw rate and lateral acceleration. Coupling also takes place with commands in the pitch channel and filtered roll rate as well. The sum is then added to the output of the ARI and filtered prior to being sent to the surface command mixer.

The surface command mixer accepts command signals from the three channels and provides software position and rate limiting (anti-wind-up). The right and left stabilator actuator commands are generated by combining the signals for elevator and differential tail (not explicitly shown in Figure 8). The signals are then position limited, differentiated, rate limited, and integrated. Similar schemes, also not shown, are used to generate the flaperon and rudder actuating commands.

The leading edge flap command is generated based on angle of attack and flight condition. Its signal path is completely separate from the three other channels, and differential left and right actuation is not possible. Prior to being sent to the actuator, the signal is limited, passed through a command servo, then limited again.

Software and mechanical breakouts which are used to protect the system from unintentional command inputs are not included in the SRF VISTA F-16 FCS, but have been added for use in this thesis. This will help to make the model more complete and accurate as well as following the direction taken in prior research (26).

3.2.3 Atmospheric Disturbances.

The SRF has three atmospheric disturbance models available for use in its simulation: gust turbulence, discrete gusts, and gust shear. In keeping with past implementations, we will focus attention on the dominant disturbance, random gust turbulence (26). The turbulence model used is based on the Dryden wind model of MIL-STD-1797A, so a discussion of the Dryden model is appropriate. This will set the foundation for the discussion of the SRF implementation as well as the development of the zero-order representation development of Section 3.3.3.

3.2.3.1 Dryden Wind Model.

The Dryden model of MIL-STD-1797A is represented in state space form as shown in Equation (19).

$$\begin{bmatrix} \dot{u}_g \\ \dot{\alpha}'_g \\ \dot{\alpha}_g \\ \dot{q}_g \\ \dot{p}_g \\ \dot{\beta}'_g \\ \dot{\beta}_g \\ \dot{r}_g \end{bmatrix} = \begin{bmatrix} \frac{-V_T}{L_u} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{-V_T}{2L_w} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & d_1 & \frac{-V_T}{2L_w} & 0 & 0 & 0 & 0 & 0 \\ 0 & d_2 & \frac{-\pi V_T^2}{8bL_w} & \frac{-\pi V_T}{4b} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-\pi V_T}{4b} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{-V_T}{2L_v} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & d_3 & \frac{-V_T}{2L_v} & 0 \\ 0 & 0 & 0 & 0 & 0 & d_4 & \frac{\pi V_T^2}{6bL_v} & \frac{-\pi V_T}{3b} \end{bmatrix} \begin{bmatrix} u_g \\ \alpha'_g \\ \alpha_g \\ q_g \\ p_g \\ \beta'_g \\ \beta_g \\ r_g \end{bmatrix} +$$

$$+ \begin{bmatrix} \sigma_u \sqrt{\frac{2V_T}{L_u}} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \sigma_w \sqrt{\frac{3}{2L_w V_T}} & 0 & 0 \\ 0 & \frac{\sigma_w \pi}{b} \sqrt{\frac{3V_T}{32L_w}} & 0 & 0 \\ 0 & 0 & \sigma_w \sqrt[n]{\frac{\pi^{10} V_T^3}{128000 b^7 L_w^2}} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \sigma_v \sqrt{\frac{3}{2L_v V_T}} \\ 0 & 0 & 0 & -\frac{\sigma_v \pi}{b} \sqrt{\frac{V_T}{6L_v}} \end{bmatrix} \begin{bmatrix} w_u \\ w_w \\ w_p \\ w_v \end{bmatrix} \qquad (19)$$

where d_1, d_2, d_3, d_4 are given by:

$$\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} \sigma_w(1-\sqrt{3})\sqrt{\frac{V_T}{8L_w^3}} \\ \frac{\sigma_w(1-\sqrt{3})\pi}{b}\sqrt{\frac{V_T^3}{128L_w^3}} \end{bmatrix} \qquad \begin{bmatrix} d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} \sigma_v(1-\sqrt{3})\sqrt{\frac{V_T}{8L_v^3}} \\ \frac{\sigma_v(1-\sqrt{3})\pi}{b}\sqrt{\frac{V_T^3}{128L_v^3}} \end{bmatrix}$$

The gust states are shown as: u_g, α'_g, α_g, q_g, p_g, β'_g, β_g, and r_g, where the subscript g denotes random turbulence, or continuous gust state. The gust variable u_g is the gust or air mass velocity effect on the aircraft forward velocity; α'_g is the air mass velocity effect on the aircraft angle of attack; α_g is the air mass velocity effect on the angle of attack; q_g is the air mass velocity effect on the aircraft pitch rate; p_g is the air mass velocity effect on the aircraft roll rate; β'_g is the air mass velocity effect on the aircraft sideslip angle; β_g is the air mass velocity effect on the sideslip angle; and r_g is the air mass velocity effect on the aircraft yaw rate. Inputs w_u, w_w, w_p, and w_v are all independent white noises of unit strength. V_T is the aircraft forward velocity, b is the wing span, σ_u, σ_v, and σ_w are turbulence RMS values, and L_u, L_v, and L_w are appropriate scale lengths. Of particular interest, are states u_g, α_g, q_g, p_g, β_g, and r_g. These have a direct influence on the aircraft state vector as discussed later in this section.

As mentioned earlier, a flight condition of .4 Mach, 20,000 ft altitude has been chosen for this thesis. Scale lengths then are chosen for the medium/high altitude condition, namely $L_u = 2L_v = 2L_w = 1750$ ft (29). Turbulence RMS values are appropriately set at $\sigma_u = \sigma_v = \sigma_w = \sigma$, where the value of σ varies to represent light, moderate, or severe turbulence categories (29). The elemental filter models developed in Section 3.3.3 assume the value $\sigma = 1$, or light turbulence. Having based the models on $\sigma = 1$, evaluations of higher and lower turbulence levels can be made by multiplying the design model equivalent discrete-time noise covariance matrix Q_d by $(\sigma')^2$. This flexibility is motivated by the subjectivity of aircraft flying qualities. Sufficient evidence exists to suggest a value of $\sigma = 1$ may more accurately describe light to moderate turbulence (29).

By relating aircraft motion to air mass motion (wind), we can understand how the Dryden disturbances enter into the aircraft equations of motion. The relationship is:

$$v_{a/i} = v_{a/m} + v_{m/i} \tag{20}$$

where subscripts a, i, and m stand for aircraft, inertial, and air mass respectively. The aircraft states directly affected then become:

$$\begin{aligned} u_s &= u - u_g \\ \alpha_s &= \alpha - \alpha_g \\ q_s &= q - q_g \\ p_s &= p - p_g \\ \beta_s &= \beta - \beta_g \\ r_s &= r - r_g \end{aligned} \tag{21}$$

where in this case, subscript s reminds us that the effect is a sum and the sign is negative by convention indicating that disturbances work against the aircraft states. Note that there is no direct affect on states θ, and ϕ, and that α_g', and β_g' have no direct effect on aircraft states.

Section 3.3.1 develops a linear, state-space model of the aircraft equations of motion. Equation (29) shows how such a model is constructed. If this model was to be incorporated into such state-space form, Equation (29) would become:

$$\dot{\alpha}(t) = Z_\theta(\theta(t) - \theta_g(t)) + Z_u(u(t) - u_g(t)) + Z_\alpha(\alpha(t) - \alpha_g(t)) + Z_q(q(t) - q_g(t)) + Z_{\delta S}\delta S(t) + Z_{\delta F}\delta F(t) \tag{22}$$

Similar equations for $\dot{u}$, $\dot{q}$, $\dot{p}$, $\dot{\beta}$, and $\dot{r}$ can also be written. If collected into state space form, the effects can be incorporated through state augmentation. The augmentation is shown in Equation

(23) below:

$$\begin{bmatrix} \dot{\theta} \\ \dot{u} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\phi} \\ \dot{\beta} \\ \dot{p} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \\ \\ \\ Homogeneous \\ Terms \\ \\ \\ \\ \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -X_u & 0 & -X_\alpha & -X_q & 0 & 0 & 0 & 0 \\ -Z_u & 0 & -Z_\alpha & -Z_q & 0 & 0 & 0 & 0 \\ -M_u & 0 & -M_\alpha & -M_q & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -Y_p & 0 & -Y_\beta & -Y_r \\ 0 & 0 & 0 & 0 & -L_p & 0 & -L_\beta & -L_r \\ 0 & 0 & 0 & 0 & -N_p & 0 & -N_\beta & -N_r \end{bmatrix} \begin{bmatrix} u_g \\ \alpha'_g \\ \alpha_g \\ q_g \\ p_g \\ \beta'_g \\ \beta_g \\ r_g \end{bmatrix} \qquad (23)$$

This state space representation will become clearer throughout the development of the state space design model in Section 3.3 below.

3.2.3.2 SRF Dryden Implementation. The SRF representation of the Dryden model only includes gust states u_g, α_g, α'_g, β_g, and β'_g. Rotational terms p_g, q_g, and r_g have been intentionally omitted by developers via the assumption that the F-16 is sufficiently small such that gusts hit all points of the aircraft at the same time. Also different from the development above, the SRF doesn't include the effects in a state space manner as shown in Equation (23). Instead, the gust effects u_g, α_g, and β_g are calculated separately, then added onto the appropriate state after dynamic equations of motion calculations. While the net effect is the same, implementation differs because the SRF aircraft model isn't in state space form.

Because rotational gust states haven't been included, the zero-order model which is developed in part in Section 3.3.3, will be incorporated into the SRF. This will form the baseline model for thesis research.

3.2.4 Input/Output. As shown in Chapter 1's Figure 1 and Chapter 2's Figure 4, the MMAE requires as its inputs, the vector of measurements $\mathbf{z}_i$ and vector of control commands $\mathbf{u}$. A FORTRAN subroutine was written which generates both vectors. The MMAE algorithm is then run from the SRF main program with a call which passes these vectors. Care was taken to make sure conversions were made where applicable, to pass variables with the correct units and trim values subtracted off where applicable to make the vectors compatible with the perturbation-based algorithm. In the case of the $\mathbf{z}_i$ vector, sensor noise had to be generated within SRF and added to corrupt the measurements appropriately. In addition to receiving the $\mathbf{u}$ and noise-corrupted $\mathbf{z}_i$ vectors, the MMAE also provides its estimation of the critical sensed variables back to the SRF VISTA's FCS. This is accomplished by replacing SRF-generated variable inputs (now sensor outputs) in the appropriate FORTRAN files with the MMAE-generated variables. These hand-offs are the only direct contact between the MMAE code and the SRF environment. There is SRF environment code used to set up the initial conditions and simulation variables prior to runtime. This is accomplished via menus at the execution of the SRF software. Appendix B contains tables which list the variable values used in this thesis.

3.2.5 Failure Insertion. Although the SRF environment contains software for failure insertion, the failures don't include all failures of interest to this thesis and only one failure can be inserted at a time. The SRF environment code was therefore modified with the logic code necessary to insert single and double combinations of the actuator and sensor failures shown in Chapter 2's Figure 4. In the case of actuators, at the time of failure, the actuating command variables are hard-coded to zero, or to their nominals if such values exist. Similarly, at the time of failure, the sensor variables are set to zero, or the nominal, plus noise. Setting variables to nominal within the SRF is sometimes necessary because the MMAE algorithm operates on perturbation equations. By setting variables within a newly written data file, the SRF software has been modified to read in failure insertion data at runtime and cut down on repetitive compiling.

3.3 Reduced Order Kalman Filter Model

The reduced-order model used in the Kalman filters is again based on the following linear, continuous-time, time-invariant dynamics equation and linear, discrete-time, time-invariant measurement equations

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{G}\mathbf{w}(t) \tag{24}$$

$$\mathbf{z}(t_i) = \mathbf{C}\mathbf{x}(t_i) + \mathbf{D}\mathbf{u}(t_i) + \mathbf{v}(t_i) \tag{25}$$

which will ultimately be transformed and discretized for use on a digital computer and be represented by

$$\mathbf{x}(t_{i+1}) = \mathbf{\Phi}\mathbf{x}(t_i) + \mathbf{B}_{\mathbf{d}}\mathbf{u}(t_i) + \mathbf{w}_d(t_i) \tag{26}$$

$$\mathbf{z}(t_i) = \mathbf{H}\mathbf{x}(t_i) + \mathbf{v}(t_i) \tag{27}$$

which are the equations used as a starting point in Chapter 2. The need for both Equations (25) and (27) will become clear in Section 3.3.2. As a reminder, **A** and **B** are the state space representations of the linearized aircraft equations of motion with **x** being the state vector and **u** the input control vector. **G** models the effect of atmospheric disturbances **w** on system states, with **w** being a zero-mean, Gaussian, white noise of strength **Q**. The sensor data **z**, is represented by a combination of states given by **H** and additional zero-mean, Gaussian, white discrete-time noise **v**, of covariance **R**. Noises **w** and **v** are assumed independent and thus uncorrelated. The following subsections describe development of each piece of the design model, beginning with Equations (24) and (25), and culminating in Equations (26) and (27).

3.3.1 Dynamics Model (A, B, Φ and B_d Matrices). The foundation upon which the reduced-order models are based is the linearized state space representation of the aircraft equations of motion:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{28}$$

The eight state and five control variables which make up **x** and **u** respectively are shown in tables 1 and 2.

Table 1. Aircraft State Vector

x	State Variables	Units
x(1)	θ pitch angle	rad
x(2)	u forward velocity	ft/sec
x(3)	α angle of attack	rad
x(4)	q pitch rate	rad/sec
x(5)	ϕ bank angle	rad
x(6)	β sideslip angle	rad
x(7)	p roll rate	rad/sec
x(8)	r yaw rate	rad/sec

Table 2. Control Vector

u	Control Variables	Units
u(1)	δS_L Left Stabilator	rad
u(2)	δS_R Right Stabilator	rad
u(3)	δF_L Left Flaperon	rad
u(4)	δF_R Right Flaperon	rad
u(5)	δR Rudder	rad

Differential equations for each of the state variables can be written in terms of dimensional derivatives expressed in the body axis. For example, aircraft angle of attack can be expressed by the following differential equation composed of dimensional derivatives, state, and control variables:

$$\dot{\alpha}(t) = Z_\theta \theta(t) + Z_u u(t) + Z_\alpha \alpha(t) + Z_q q(t) + Z_{\delta S} \delta S(t) + Z_{\delta F} \delta F(t) \tag{29}$$

In like manner, all eight state variables can be represented. Their equations are collected in convenient state space form and become:

$$
\begin{bmatrix} \dot{\theta} \\ \dot{u} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\beta} \\ \dot{\phi} \\ \dot{p} \\ \dot{r} \end{bmatrix}
=
\begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
X_\theta & X_u & X_\alpha & X_q & 0 & 0 & 0 & 0 \\
Z_\theta & Z_u & Z_\alpha & Z_q & 0 & 0 & 0 & 0 \\
M_\theta & M_u & M_\alpha & M_q & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & \Phi_r \\
0 & 0 & 0 & 0 & Y_\phi & Y_\beta & Y_p & Y_r \\
0 & 0 & 0 & 0 & 0 & L_\beta & L_p & L_r \\
0 & 0 & 0 & 0 & 0 & N_\beta & N_p & N_r
\end{bmatrix}
\begin{bmatrix} \theta \\ u \\ \alpha \\ q \\ \beta \\ \phi \\ p \\ r \end{bmatrix}
+
$$

$$
+
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
X_{\delta S} & 0 & X_{\delta F} & 0 & 0 \\
Z_{\delta S} & 0 & Z_{\delta F} & 0 & 0 \\
M_{\delta S} & 0 & M_{\delta F} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & Y_{\delta DS} & 0 & Y_{\delta DF} & Y_{\delta_r} \\
0 & L_{\delta DS} & 0 & L_{\delta DF} & L_{\delta_r} \\
0 & N_{\delta DS} & 0 & N_{\delta DF} & N_{\delta_r}
\end{bmatrix}
\begin{bmatrix} \delta S \\ \delta DS \\ \delta F \\ \delta DF \\ \delta R \end{bmatrix}
\qquad (30)
$$

The values for the dimensional derivatives of Equation (30) are obtained at a chosen trim condition using the SRF VISTA simulation. For this thesis, a trim condition of 0.4 Mach, 20,000 ft altitude was chosen to facilitate comparison with past research. The **A** used in this research which comes from Equation (30) and the SRF VISTA simulation is (to four digits):

$$
\mathbf{A} =
\begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
-31.7298 & -0.0051 & 33.1831 & -70.0651 & 0 & 0 & 0 & 0 \\
-0.0133 & -0.0002 & -0.4007 & 0.9970 & 0 & 0 & 0 & 0 \\
0.0007 & -0.0015 & 1.1742 & -0.5302 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.1728 \\
0 & 0 & 0 & 0 & 0.0773 & -0.1104 & 0.1715 & -0.9977 \\
0 & 0 & 0 & 0 & 0 & -18.1336 & -1.5880 & 0.0321 \\
0 & 0 & 0 & 0 & 0 & 2.9112 & -0.0426 & -0.2817
\end{bmatrix}
\qquad (31)
$$

The **B** however, isn't as as straightforward **A**. Equation (30)'s control inputs are in terms of δS, δF, δDS, and δDF, not the required δS_R, δS_L, δF_R, and δF_L. To fix this, the stabilator and flaperon control authority must be divided between the left and right surfaces. For longitudinal deflections δS and δF can be expressed as $\frac{1}{2}X_{\delta S}(\delta S_L + \delta S_R)$ and $\frac{1}{2}X_{\delta F}(\delta F_L + \delta F_R)$ respectively. For lateral, differential deflections (note the sign change) δDS and δDF become $\frac{1}{2}X_{\delta DS}(-\delta S_L + \delta S_R)$ and $\frac{1}{2}X_{\delta DF}(-\delta F_L + \delta F_R)$ respectively. Making these changes, **B** becomes

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ .5X_{\delta S} & .5X_{\delta S} & .5X_{\delta F} & .5X_{\delta F} & 0 \\ .5Z_{\delta S} & .5Z_{\delta S} & .5Z_{\delta F} & .5Z_{\delta F} & 0 \\ .5M_{\delta S} & .5M_{\delta S} & .5M_{\delta F} & .5M_{\delta F} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -.5Y_{\delta DS} & .5Y_{\delta DS} & -.5Y_{\delta DF} & .5Y_{\delta DF} & Y_{\delta R} \\ -.5L_{\delta DS} & .5L_{\delta DS} & -.5L_{\delta DF} & .5L_{\delta DF} & L_{\delta R} \\ -.5N_{\delta DS} & .5N_{\delta DS} & -.5N_{\delta DF} & .5N_{\delta DF} & N_{\delta R} \end{bmatrix} \tag{32}$$

For this thesis, at .4 Mach, 20,000 ft, **B** becomes

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1.0667 & 1.0667 & -0.1625 & -0.1625 & 0 \\ -0.0343 & -0.0343 & -0.0102 & -0.0102 & 0 \\ -1.8416 & -1.8416 & 0.0902 & 0.0902 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -0.0069 & 0.0069 & -0.0003 & -0.0003 & 0.0170 \\ 4.5683 & -4.5683 & 6.2649 & -6.2649 & 2.8853 \\ 0.5318 & -0.5318 & 0.0686 & -0.0686 & -1.1784 \end{bmatrix} \tag{33}$$

This completes our basic representation of the aircraft equations of motion. Actuator dynamics need also be taken into account however, and then augmented to the **A** and **B** matrices just developed.

Typically, actuators are modeled with fourth order transfer functions. From VISTA F-16 Drawing 711DFCS007, the transfer function used to describe left and right stabilators and flaperons as well as the rudder is (6):

$$T(s) = \frac{\delta C}{\delta C_{CMD}} = \frac{(20.2)(141.4)(5214.5)}{(s + 20.2)(s + 141.4)(s^2 + 107.0s + 5214.5)} \tag{34}$$

where δC is the actuator position and δC_{CMD} the commanded position. For simplicity and to keep the order of the filter models down, a first order actuator model is used. The reduced order representation of the fourth order model retains the dominant pole effects only and is shown below:

$$T(s) = \frac{20.2}{s + 20.2} \tag{35}$$

Previous research efforts have enjoyed success using the first order model (26,33). However, in those theses, the first order model was used in *both* the design model and truth model. Because the SRF contains a nonlinear, fourth order, ISA model, a Bode analysis was performed to verify the validity of our reduced order model, Equation (35).

Figure 9 shows a graphic representation of this analysis. Bode plots for the fourth order transfer function, Equation (34) from Drawing 711DFCS007, and the first order representation, Equation (35), are superimposed on the Bode plot analysis along with a better first order fit over the frequencies of interest (0 to 20 rad/s). Based on this analysis, the reduced order model used in the past is replaced in this thesis by

$$T(s) = \frac{19}{s + 19} \tag{36}$$

Using the rudder as an example, the first order transfer functions can again be written in differential equation form,

$$\delta \dot{R} = -19\delta R + 19\delta R_{CMD}$$

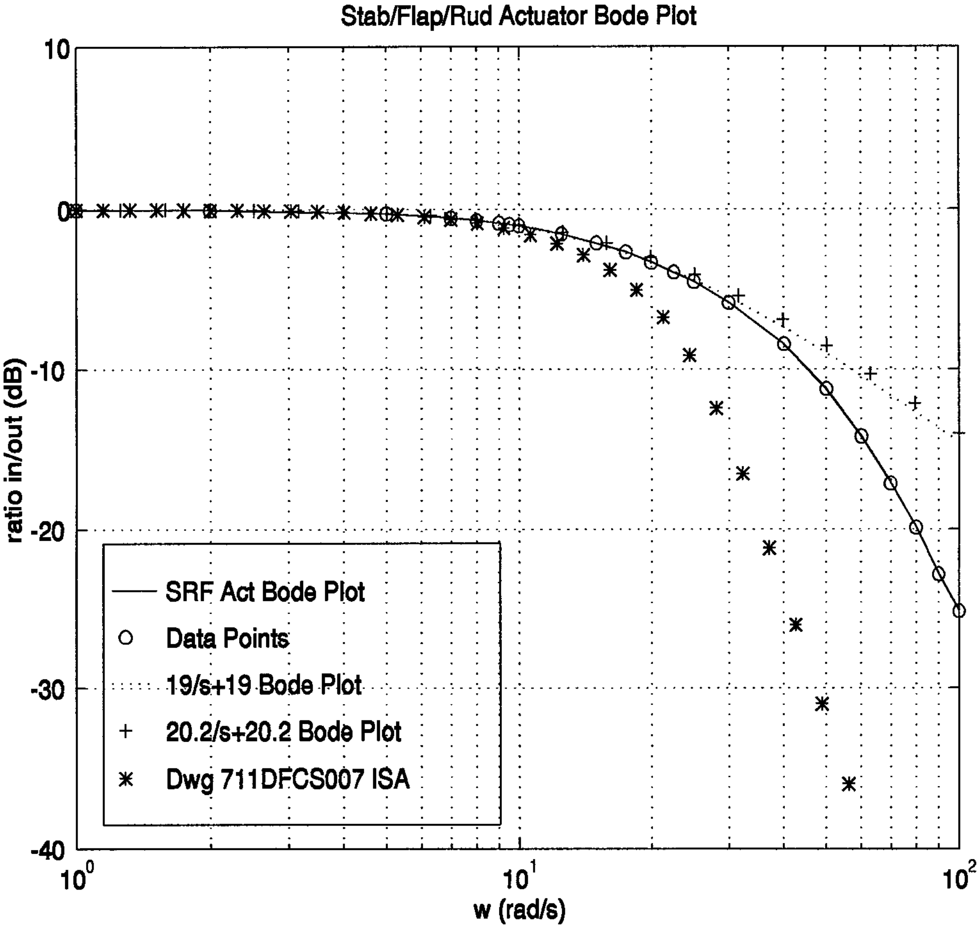


Figure 9. ISA Bode Analysis

and then collected in state space form:

$$\begin{bmatrix} \delta\dot{S}_L \\ \delta\dot{S}_R \\ \delta\dot{F}_L \\ \delta\dot{F}_R \\ \delta\dot{R} \end{bmatrix} = \begin{bmatrix} -19 & 0 & 0 & 0 & 0 \\ 0 & -19 & 0 & 0 & 0 \\ 0 & 0 & -19 & 0 & 0 \\ 0 & 0 & 0 & -19 & 0 \\ 0 & 0 & 0 & 0 & -19 \end{bmatrix} \begin{bmatrix} \delta S_L \\ \delta S_R \\ \delta F_L \\ \delta F_R \\ \delta R \end{bmatrix} +$$

$$+\begin{bmatrix} 19 & 0 & 0 & 0 & 0 \\ 0 & 19 & 0 & 0 & 0 \\ 0 & 0 & 19 & 0 & 0 \\ 0 & 0 & 0 & 19 & 0 \\ 0 & 0 & 0 & 0 & 19 \end{bmatrix} \begin{bmatrix} \delta S_{L_{CMD}} \\ \delta S_{R_{CMD}} \\ \delta F_{L_{CMD}} \\ \delta F_{R_{CMD}} \\ \delta R_{CMD} \end{bmatrix} \tag{37}$$

This state space representation is then augmented to Equation (28), forming the new equation

$$\dot{\mathbf{x}}_{\mathbf{aug}}(t) = \mathbf{A}_{\mathbf{aug}}\mathbf{x}_{\mathbf{aug}}(t) + \mathbf{B}_{\mathbf{aug}}\mathbf{u}_{\mathbf{new}}(t) \tag{38}$$

The augmented state vector and new control vector are again displayed in tabular form in Tables 3 and 4.

Table 3. Augmented Aircraft State Vector

x	State Variables	Units
x(1)	θ pitch angle	rad
x(2)	u forward velocity	ft/sec
x(3)	α angle of attack	rad
x(4)	q pitch rate	rad/sec
x(5)	ϕ bank angle	rad
x(6)	β sideslip angle	rad
x(7)	p roll rate	rad/sec
x(8)	r yaw rate	rad/sec
x(9)	δS_L left stabilator position	rad
x(10)	δS_R right stabilator position	rad
x(11)	δF_L left flaperon position	rad
x(12)	δF_R right flaperon position	rad
x(13)	δR rudder position	rad

Table 4. New Control Vector

u	Control Variables	Units
u(1)	$\delta S_{L_{CMD}}$ Left Stabilator Command	rad
u(2)	$\delta S_{R_{CMD}}$ Right Stabilator Command	rad
u(3)	$\delta F_{L_{CMD}}$ Left Flaperon Command	rad
u(4)	$\delta F_{R_{CMD}}$ Right Flaperon Command	rad
u(5)	δR_{CMD} Rudder Command	rad

For completeness, the correct augmentations of $\mathbf{A}_{\mathbf{aug}}$ and $\mathbf{B}_{\mathbf{aug}}$ are shown below.

$$\mathbf{A_{aug}} = \begin{bmatrix} [\mathbf{A}] & [\mathbf{B}] \\ [0] & \begin{bmatrix} -19 & 0 & 0 & 0 & 0 \\ 0 & -19 & 0 & 0 & 0 \\ 0 & 0 & -19 & 0 & 0 \\ 0 & 0 & 0 & -19 & 0 \\ 0 & 0 & 0 & 0 & -19 \end{bmatrix} \end{bmatrix} \tag{39}$$

$$\mathbf{B_{aug}} = \begin{bmatrix} [0] \\ \begin{bmatrix} 19 & 0 & 0 & 0 & 0 \\ 0 & 19 & 0 & 0 & 0 \\ 0 & 0 & 19 & 0 & 0 \\ 0 & 0 & 0 & 19 & 0 \\ 0 & 0 & 0 & 0 & 19 \end{bmatrix} \end{bmatrix} \tag{40}$$

The final step in this section takes the continuous $\mathbf{A_{aug}}$ and $\mathbf{B_{aug}}$ matrices into the discrete-time domain. Of course, this is to facilitate computer implementation of the algorithm. Using the Pade approximation method of time domain transfer, the $\mathbf{A_{aug}}$ and $\mathbf{B_{aug}}$ matrices are transformed into $\mathbf{\Phi}$ and $\mathbf{B_d}$. In this thesis, these matrices are given in Appendix A.

3.3.2 Measurement Model (C, D, and H Matrices). The next logical piece of the design model to develop is the measurement model **H** matrix. As a starting point, the linear, discrete, time-invariant generalized measurement equation is written below:

$$\mathbf{z}(t_i) = \mathbf{C}\mathbf{x}(t_i) + \mathbf{D}\mathbf{u}(t_i) \tag{41}$$

where **z** is the vector of sensed variables. In general, when considering sensor data, sampled-data measurements are quite common, thus no future discretization will be necessary. Also, for the time being, we'll assume noiseless measurements. Uncertainty within the dynamics and measurement equations will be added to the model in the next section.

As Equation (41) suggests, the vector of sensed variables can be expressed as a linear combination of state and control variables. (Note that no subscripts on **x** and **u** indicate Tables 1 and 2 apply at this point.) The seven flight critical sensor variables which make up **z** are shown in Table 5. These variables are considered flight critical because they are directly fed back to, and required by, the F-16's Block-40 FCS. Clearly u,α, q, p, and r are already available as states. A_n and A_y are not as straightforward. These variables must be generated as a linear combination of states and controls.

Table 5. Sensor Variables Vector

z	Sensor Variables	Units
z(1)	u forward velocity	ft/sec
z(2)	α angle of attack	rad
z(3)	q pitch rate	rad/sec
z(4)	A_n normal acceleration (cg)	g's
z(5)	p roll rate	rad/sec
z(6)	r yaw rate	rad/sec
z(7)	A_y lateral acceleration (cg)	g's

The variables A_n and A_y can be expressed in state variable form as

$$A_{n_{cg}} = -\frac{\bar{U}}{32.2}(\dot{\alpha} - q) \tag{42}$$

and

$$A_{y_{cg}} = \frac{\bar{U}}{32.2}(\dot{\beta} + r) - \sin\phi \tag{43}$$

where $\bar{U}$ is the nominal forward velocity and 32.2 is the conversion factor from $\frac{ft}{sec^2}$ to g's. Equation (43) contains the nonlinear term $\sin\phi$. Under the small angle, perturbation equation assumption, this term can be linearized to ϕ in the A_y equation. In order to generate these variables, the differential equation relationships for $\dot{\alpha}$ and $\dot{\beta}$ must be obtained from Section 3.3.1. Equation (29) shows the desired relationship for $\dot{\alpha}$. Rewritten in terms of the correct control input variables, $\dot{\alpha}$

and $\dot{\beta}$ become:

$$\begin{aligned}\dot{\alpha} \quad &= \quad Z_\theta\theta + Z_u u + Z_\alpha\alpha + Z_q q + \\ &\quad .5Z_{\delta S}\delta S_L + .5Z_{\delta S}\delta S_R + .5Z_{\delta F}\delta F_L + .5Z_{\delta F}\delta F_L \end{aligned} \tag{44}$$

$$\begin{aligned}\dot{\beta} \quad &= \quad Y_\phi\phi + Y_\beta\beta + Y_p p + Y_r r - \\ &\quad .5Y_{\delta DS}\delta S_L + .5Y_{\delta DS}\delta S_R - .5Y_{\delta DF}\delta F_L + .5Y_{\delta DF}\delta F_R + Y_{\delta R}\delta R \end{aligned} \tag{45}$$

With these relationships established, Equations (42) and (43) can be calculated. Equation (41) can then be conveniently written down as:

$$\begin{aligned}\begin{bmatrix} u \\ \alpha \\ q \\ A_n \\ p \\ r \\ A_y \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{\bar{U}}{32.2}[Z_\theta & Z_u & Z_\alpha & Z_q - 1 & 0 & 0 & 0 & 0] \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & [\frac{\bar{U}}{32.2}Y_\phi - 1] & \frac{\bar{U}}{32.2}[Y_\beta & Y_p & Y_r + 1] \end{bmatrix} \begin{bmatrix} \theta \\ u \\ \alpha \\ q \\ \phi \\ \beta \\ p \\ r \end{bmatrix} \\ &+ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\frac{\bar{U}}{32.2}[.5Z_{\delta S} & .5Z_{\delta S} & .5Z_{\delta F} & .5Z_{\delta F} & 0] \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{\bar{U}}{32.2}[-.5Y_{\delta DS} & .5Y_{\delta DS} & -.5Y_{\delta DF} & .5Y_{\delta DF} & Y_{\delta R}] \end{bmatrix} \begin{bmatrix} \delta S_L \\ \delta S_R \\ \delta F_L \\ \delta F_R \\ \delta R \end{bmatrix} \end{aligned} \tag{46}$$

Now, in order to make this equation compatible with the augmented system developed in the previous section, it too must be augmented. Having done this, the measurement equation is in the form required in Chapter 2, Equation (4). The correct augmentation is demonstrated below.

$$\mathbf{H} = \begin{bmatrix} [\mathbf{C}] & [\mathbf{D}] \end{bmatrix} \tag{47}$$

The values used in this thesis for **C** and **D**, which become **H** are (to four figures)

$$\mathbf{H} = \left[\begin{array}{ccccccccc}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots \\
0.1728 & 0.0027 & 5.1872 & 0.0393 & 0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \cdots \\
0 & 0 & 0 & 0 & 1 & -1.4290 & 2.2202 & 0.0303 & \cdots
\end{array}\right.$$

$$\left.\begin{array}{cccccc}
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0.4438 & 0.4438 & 0.1326 & 0.1326 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & -0.0895 & 0.0895 & -0.0039 & 0.0039 & 0.2198
\end{array}\right] \tag{48}$$

3.3.3 Noises (R, G, and Q Matrices). The final pieces of the design model are dynamics and measurement equation uncertainties. The measurement uncertainly quite straightforwardly represents noise in sensor data. Based upon past research, the bandwidth of sensor noise is very large compared to the frequency response of the aircraft. Under this condition, we can conclude that the noise is essentially white. What needs to be determined then, are the variances of the seven noises which corrupt the sampled-data vector of sensed variables of Table 5.

Several research efforts have provided sensor noise variances which were determined using F-8 flight test data (18,26,32,33). For the most part, these numbers were quite conservative, possibly to a fault because they limited performance capabilities of the algorithm. This thesis will employ the variances which were actually used in (26–28), as opposed to those documented in past write-ups. Also, this thesis will continue with the assumption that all seven noises are uncorrelated. This is perhaps a bit specious, but to the level of complexity desired, independent noises will suffice. To four figures, **R** is:

$$\mathbf{R} = \begin{bmatrix} 4.8 \times 10^{-6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.6 \times 10^{-5} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3.6 \times 10^{-5} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 \times 10^{-4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4.0 \times 10^{-4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3.6 \times 10^{-5} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2.5 \times 10^{-5} \end{bmatrix} \begin{bmatrix} (ft/sec)^2 \\ rad^2 \\ (rad/sec)^2 \\ (ft/sec)^2 \\ (rad/sec)^2 \\ (rad/sec)^2 \\ (ft/sec)^2 \end{bmatrix} \tag{49}$$

With this noise in place, it can be added to the results of Section 3.3.2, forming the complete design model measurement Equation (4).

The dynamics equation uncertainty is a result of unpredictable atmospheric disturbances. These disturbances are documented in MIL-STD-1797A and the Dryden wind model (29). The Dryden model uses shaping filters driven by Gaussian white noise to represent atmospheric input effects on system state. Our design model representation uses a zero-order approximation to the Dryden model which has been developed in past research (18). A zero-order approximation is again used to keep the order of the elemental filters to a minimum while still retaining the dominant effects of the disturbances.

The reduced order model is generated by approximating the six time-correlated wind disturbances of the Dryden model with six white noises. The strengths of the white noise approximations are calculated by averaging the Dryden model power spectral densities over their appropriate frequency ranges (18). MIL-STD-1797A provides the Dryden power spectral densities. Their average magnitudes from 0 $\frac{rad}{sec}$ to the aircraft bandwidth of interest provides the approximation strengths. Table 6 contains the vector of white noises along with the corresponding Dryden disturbances which they approximate.

Table 6. Vector of White Noises

x	Approx	Disturbances
w(1)	w_{u_g}	u_g forward velocity
w(2)	w_{α_g}	α_g angle of attack
w(3)	w_{q_g}	q_g pitch rate
w(4)	w_{p_g}	p_g roll rate
w(5)	w_{β_g}	β_g sideslip angle
w(6)	w_{r_g}	r_g yaw rate

Since $\alpha_g(t)$ and $q_g(t)$ are both functions of $w_{\alpha_g}(t)$, and $\beta_g(t)$ and $r_g(t)$ are both functions of w_{β_g}, the cross spectral densities $\Phi_{\alpha_g q_g}(s)$ and $\Phi_{\beta_g r_g}(s)$ must be included in the model. Equation (50) shows the resulting **Q** which must be developed.

$$\mathbf{Q} = \begin{bmatrix} Q_{u_g} & 0 & 0 & 0 & 0 & 0 \\ 0 & Q_{\alpha_g} & Q_{\alpha_g q_g} & 0 & 0 & 0 \\ 0 & Q_{\alpha_g q_g} & Q_{q_g} & 0 & 0 & 0 \\ 0 & 0 & 0 & Q_{p_g} & 0 & 0 \\ 0 & 0 & 0 & 0 & Q_{\beta_g} & Q_{\beta_g r_g} \\ 0 & 0 & 0 & 0 & Q_{\beta_g r_g} & Q_{r_g} \end{bmatrix} \tag{50}$$

The bandwidths for aircraft states and associated noise strengths are based upon a wind turbulence RMS value of $\sigma = 1\frac{ft}{sec}$. These are listed in Table 7 (26).

Table 7. Zero-Order White Noise Strengths

Variable	Aircraft Bandwidth	Units	Average Noise Strength	Units
u	0.25	$\frac{rad}{sec}$	4.5×10^{-1}	$\frac{ft^2}{rad \cdot sec}$
α	4.0	$\frac{rad}{sec}$	3.0^{-6}	$rad \cdot sec$
q	20.0	$\frac{rad}{sec}$	1.5×10^{-6}	$\frac{rad}{sec}$
α vs q	4.0	$\frac{rad}{sec}$	1.1×10^{-8}	rad^2
p	15.0	$\frac{rad}{sec}$	6.0×10^{-6}	$\frac{rad}{sec}$
β	3.5	$\frac{rad}{sec}$	3.0×10^{-6}	$rad \cdot sec$
r	7.0	$\frac{rad}{sec}$	2.4×10^{-6}	$\frac{rad}{sec}$
β vs r	3.5	$\frac{rad}{sec}$	6.3×10^{-9}	rad^2

Since each of the power spectral density equations contains a σ^2 term, which indicates the RMS strength of wind turbulence, variability of the turbulence values can be achieved by multiplying σ^2 by an appropriate scalar k, resulting in a new turbulence σ'^2 ($\sigma'^2 = k\sigma^2$). Using the information contained in Table 7, the **Q** used in this thesis then becomes

$$\mathbf{Q} = \begin{bmatrix} 4.5 \times 10^{-2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.0 \times 10^{-6} & 1.1 \times 10^{-8} & 0 & 0 & 0 \\ 0 & 1.1 \times 10^{-8} & 1.5 \times 10^{-6} & 0 & 0 & 0 \\ 0 & 0 & 0 & 6.0 \times 10-6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3.0 \times 10-6 & 6.3 \times 10^{-9} \\ 0 & 0 & 0 & 0 & 6.3 \times 10^{-9} & 2.4 \times 10^{-6} \end{bmatrix} \tag{51}$$

The final consideration is how the approximated noises are brought into the dynamics equation, or how they affect the system states. **G** remains to be developed.

Comparing the order of states in the state vector given in Table 1 and the white noise vector in Table 6, one notices the mismatch at β and p. This switch necessitates a reversal of the sixth and seventh rows of **G** as shown below.

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{52}$$

The white noises are incorporated into the aircraft equations of motion by multiplying the **G** matrix by the aircraft dimensional derivatives as shown in Equation (23) of Section 3.2.3.1. (Note that α'_g and β'_g are not a part of this representation, thus their columns are omitted.) This yields the design model **G**:

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ X_u & X_\alpha & X_q & 0 & 0 & 0 \\ Z_u & Z_\alpha & Z_q & 0 & 0 & 0 \\ M_u & M_\alpha & M_q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Y_p & Y_\beta & Y_r \\ 0 & 0 & 0 & L_p & L_\beta & L_r \\ 0 & 0 & 0 & N_p & N_\beta & N_r \end{bmatrix} \tag{53}$$

where the dimensional derivatives are as defined in Section 3.3.1. For completeness, the **G** used in this thesis becomes:

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0051 & 33.1831 & -70.0651 & 0 & 0 & 0 \\ -0.0002 & -0.4007 & 0.9970 & 0 & 0 & 0 \\ -0.0015 & 1.1742 & -0.5302 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1715 & -0.1104 & -0.9977 \\ 0 & 0 & 0 & -1.5880 & -18.1336 & 0.0321 \\ 0 & 0 & 0 & -0.0426 & 2.9112 & -0.2817 \end{bmatrix} \tag{54}$$

The last step necessary considers appropriate discretization of the noises for sampled-data operation. The matrix of covariances **R** is already in discrete-time form, so nothing further is done in update Equation (27). The sampled-data propagation Equation (26) still requires development of a discrete-time representation of its uncertainty. This is done by duplicating the statistics of the discretized continuous process noise with an equivalent discrete-time white Gaussian noise. The result is the zero-mean, white, Gaussian noise $\mathbf{w}_d$ of strength $\mathbf{Q}_d$ which is calculated in Appendix A.

For a more complete development of the zero-order approximation design model, see (18).

3.3.4 Failure Modeling.

3.3.4.1 Actuator Failure Simulation.

There are a number of ways in which an actuator could fail: stuck to zero, stuck at present deflection, failed to free stream, etc. For simplicity and compatibility with past research, this thesis assumes stuck to zero. That means complete loss of actuator control with zero degrees of perturbed deflection. In order to model actuator failures within the design model, the column associated with the failed actuator in **B** is replaced by all zeros.

Another way in which actuators could fail is partial loss of control authority, or a soft failure. To model this, arbitrary percentages of the affected column on **B** are selected. Research with linear truth models has shown the MMAE algorithm capable of blending the appropriate percentages of hard failure and fully functional filter outputs to accommodate the soft failure (26). Soft actuator failures will not be further pursued in this thesis.

3.3.4.2 Sensor Failure Simulation.

Sensor failures which occur in this thesis are assumed to be a complete loss of measured output, or simply noise only coming from the failed sensor. In order to model sensor failures within the design model, the row associated with the failure in **H** is replaced by all zeros. Soft failures in the sensors are also of concern. A sudden increase in sensor noise has been modeled in the past by an appropriate step increase in measurement noise variances within the truth model. Measurement biases have also been modeled within the truth model. The MMAE has shown limited success in detecting soft sensor failures. Again, soft failures will not be examined within the nonlinear environment in this thesis.

3.3.4.3 Multiple Failures.

Multiple failures include all combinations of hard actuator and sensor failures taken two at a time. The design model naturally assumes some combination of zeroed **B** columns and **H** rows to model dual failures. As discussed in Chapter 2, Section 2.2.2.3, a hierarchical structure is used to accommodate the 13 banks required by this thesis. It is one of the goals of this thesis to enhance the algorithm's multiple failure detection and isolation capability.

3.4 Chapter Summary

This chapter has developed, in detail, the models used in this thesis. The truth model is different from research efforts in the past in that it is a fully nonlinear, six-degree-of-freedom, aircraft simulation not designed specifically for the MMAE algorithm. The design models used within the Kalman filters are essentially the same as what has been used in the past. The goal of this thesis is to run the MMAE against the new truth model and demonstrate its capabilities. Methods for handling failures were discussed as well. Finally, for repeatability, a step-by-step procedure for generating the MMAE algorithm is presented in Appendix D. The next section will present results from research based on these models.

4. Results

4.1 Chapter Overview

The following chapter provides the results of this thesis research effort. The layout follows directly from the research questions posed in Chapter 1. First we address probability convergence, then full envelope coverage. Under probability convergence, we spend a good deal of time detailing the single failure case and performance enhancement techniques explored. Also, results of the dual failure cases are provided and characterized. Soft failures were omitted from the study due to sponsor preference to investigate full envelope coverage instead, but are also discussed somewhat. Where appropriate, at the end of each section, the research questions posed in Chapter 1 are answered directly. The chapter then concludes with a summary.

4.2 Probability Convergence

The first area of research interest had to do with probability convergence. In the SRF VISTA simulation environment, we wished to see if the MMAE elemental filters converged to the correct failure hypothesis. In other words, we wanted to determine if the filter probability whose hypothesis was based on the correct failure condition approached unity (0.988) over the period of failure. The following subsections discuss the single and double failure scenarios.

4.2.1 Single Failures. The bulk of this research effort centered on the single failure scenario. As somewhat expected, there were more difficulties associated with a higher order and fully nonlinear truth model implementation than will the simpler environments used in previous studies (9,14,26). The following sections attempt to show a meaningful progression from where the research began, through some of the solvable difficulties, to the conclusion. Focus is on the techniques used to improve performance rather than on final performance levels. As will become apparent along the way, there are numerous tradeoffs and many techniques to consider, all of which have their costs and benefits. Due to the somewhat arbitrary nature of this research exercise (there were no explicit specifications provided at the outset), it became frustratingly difficult to define "adequate" performance and not go overboard trying to tune the MMAE accordingly. Ultimately, performance specifications must be in place to define "adequate" performance and an MMAE designed appropriately. As such, this section demonstrates performance improvement, the techniques responsible for the improvement, and the costs associated with the techniques.

4.2.1.1 Replication of Past Research. As a starting point, this research sought to replicate, as closely as possible, the results of Menke (26–28). As a reminder, he also researched performance of the MMAE implemented on a VISTA F-16, but his aircraft truth model was linearized and of equal order as his design model. Our research utilized a modified version of the SRF VISTA, nonlinear simulation and a stand-alone version of the Menke MMAE, written by Hanlon and refined by Lane (9,14). Even so, the original idea was to follow the approach taken by Menke. To that end, a filter generation routine more accurate yet based on the old one was written and used, and the same dithering scheme was employed. Significant unavoidable differences, though, included different nominal **A** and **B** matrices generated by SRF and a different actuator model used in the reduced order design model. Both deserve further discussion.

One small, but notable difference between this simulation and Menke's begins with the linearized trim **A** and **B** matrices. Menke used a software package called GENESIS and this research used SRF VISTA's internal linearization routine (2). The result was two different sets of trim equations for the same trim condition. Most of the matrix entries are the same or very close, but a few are not. Also, the SRF VISTA package **B** matrix did not include the leading edge flap effects, which made it 8×5 instead of 8×6. This effects an involuntary reduction of order at the outset. Since the Genesis tool was unavailable, an in-depth exploration of these differences was impossible. We chose to proceed, however, with the equations provided by SRF VISTA, as that would ultimately be the host environment. It is duly noted though, that the MMAE's basic performance quality is directly dependent on the accuracy of its reduced order equations of motion.

While performance enhancement techniques such as those forthcoming can minimize the effect of errors in the equations of motion (EOM) to a large extent, care should be taken at the beginning of every future implementation to obtain the most accurate EOM.

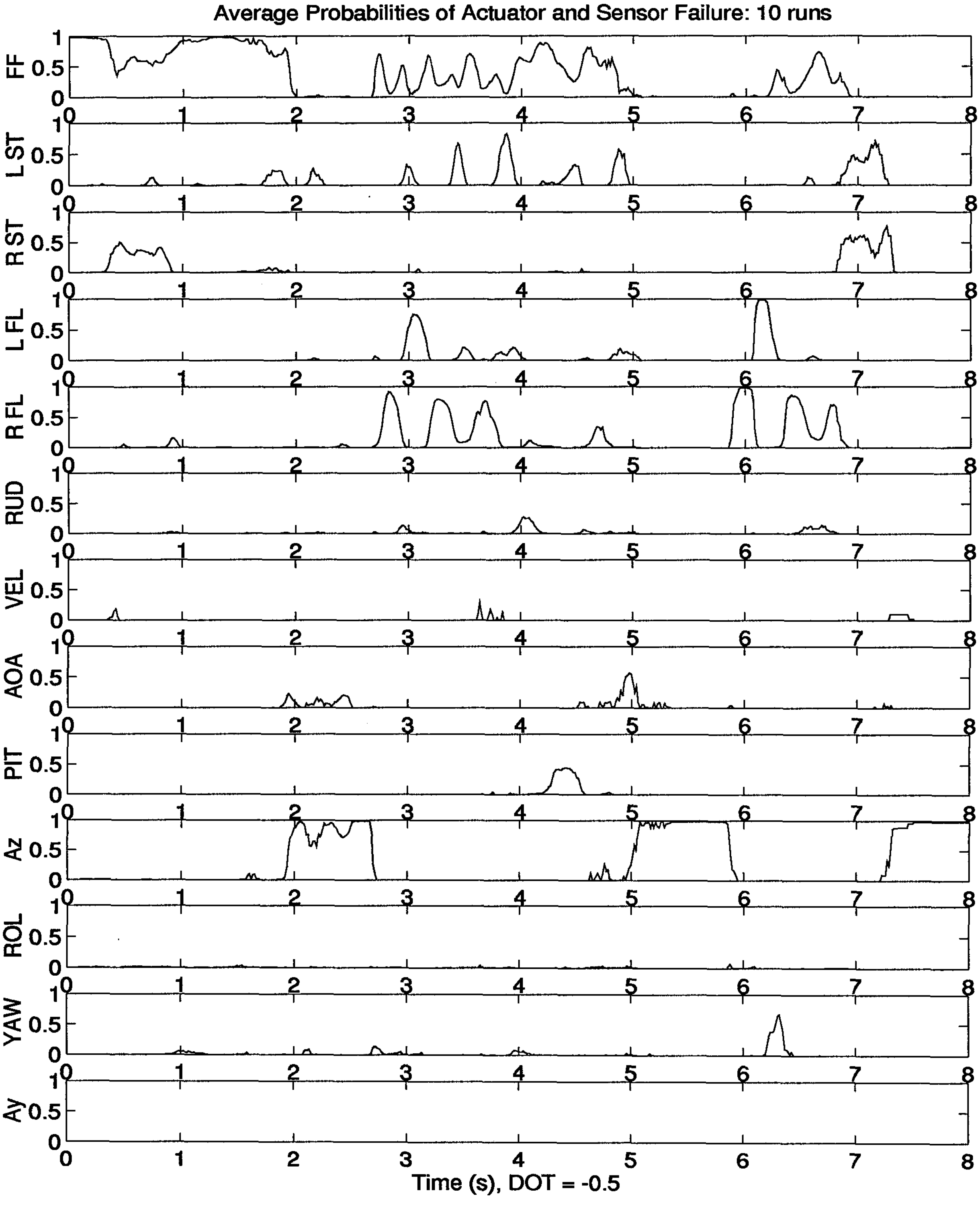
The other important difference is in the reduced order design actuator model. This is a continuation of the discussion in Chapter 3. In Chapter 3, we recognized that the SRF actuators were simulated differently than in the past. We also attempted to characterize them by performing a rough Bode analysis, the result of which placed the first order pole at 19.0 $\frac{rad}{s}$. Subsequent empirical evidence consistently showed better performance when the pole was placed at 14.0 $\frac{rad}{s}$ instead. While our Bode analysis doesn't lead to this conclusion, there is some support to the possibility of the closer pole effecting better performance by acting as a low pass filter of sorts. The predominant effect of moving the pole location to the lower frequency value was to minimize existing dropouts in probabilities, and probability dropouts are what has been attributed to higher frequency effects in the past (9). Additionally, it is quite possible that a pole at 14.0 $\frac{rad}{s}$ is simply a better match to the SRF VISTA actuator simulation and the Bode analysis was flawed or inadequate due to nonlinearities. At any rate, again it is important to note the need for an accurate model. In this case, we are striving to ensure a best match to the SRF VISTA simulation rather than to an actual F-16, because the test is taking place in a computer. Adjustments can accordingly take place from implementation to implementation. In our research then, the design model, first order representation of the actuators became:

$$T(s) = \frac{14.0}{s + 14.0} \tag{55}$$

With the above differences noted, we proceeded to investigate the MMAE's ability to detect failures in the presence of a higher order truth model. Figures 10 to 22 show the results. Again as a reminder, this style of plot depicts the averages of 13 filters running in parallel for 8 seconds over 10 Monte Carlo runs. For each is plotted its probability of being the correct failure hypothesis. A single failure is induced at 3 seconds. Moreover, the nomenclature is: **FF** = fully functional, or no-failure condition; **L ST** = left stabilator actuator; **R ST** = right stabilator actuator; **L FL** = left flaperon actuator; **R FL** = right flaperon actuator; **RUD** = rudder actuator; **VEL** = forward velocity sensor; **AOA** = angle of attack sensor; **PIT** = pitch rate sensor; **Az** = normal acceleration sensor; **ROL** = roll rate sensor; **YAW** = yaw rate sensor; **Ay** = lateral acceleration sensor.

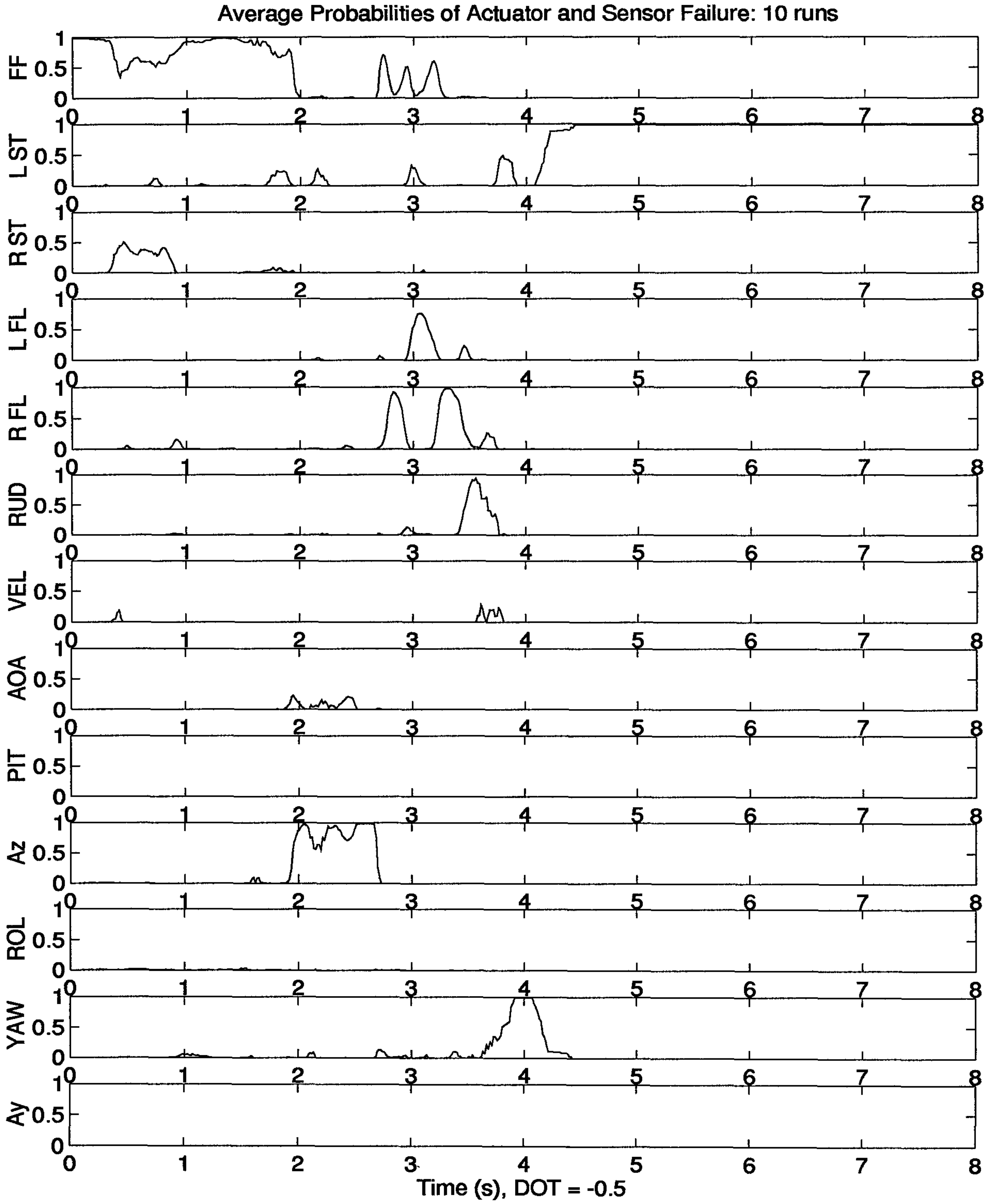
The results clearly show us a few things. First, performance is poor across the board. There are unacceptably large and numerous probability dropouts from the correct failure hypothesis. Figure 10, the fully functional case, exemplifies the problems. Over the eight second period, numerous false alarms might be declared. A second but important point to notice is that, in all cases (Figs. 10 to 22), the MMAE is at least struggling to declare the correct failure. Indications are that the correct channel is being excited for some period of time during each failure run. The conclusion which can be drawn from the observations above is that our reduced order design models are close, but suffer from the omission of higher order effects now included in the truth model. Clearly these results reflect one of the quite possible outcomes predictable prior to experimentation: namely, that a reduced order design model without tuning will indeed struggle against a higher order truth model. These effects had yet to be observed, but were certainly anticipated.

The task at hand then, is to acknowledge the uncertainty in our models and go about tuning for improved performance if possible. The next section describes various techniques attempted, their results, and the costs associated.



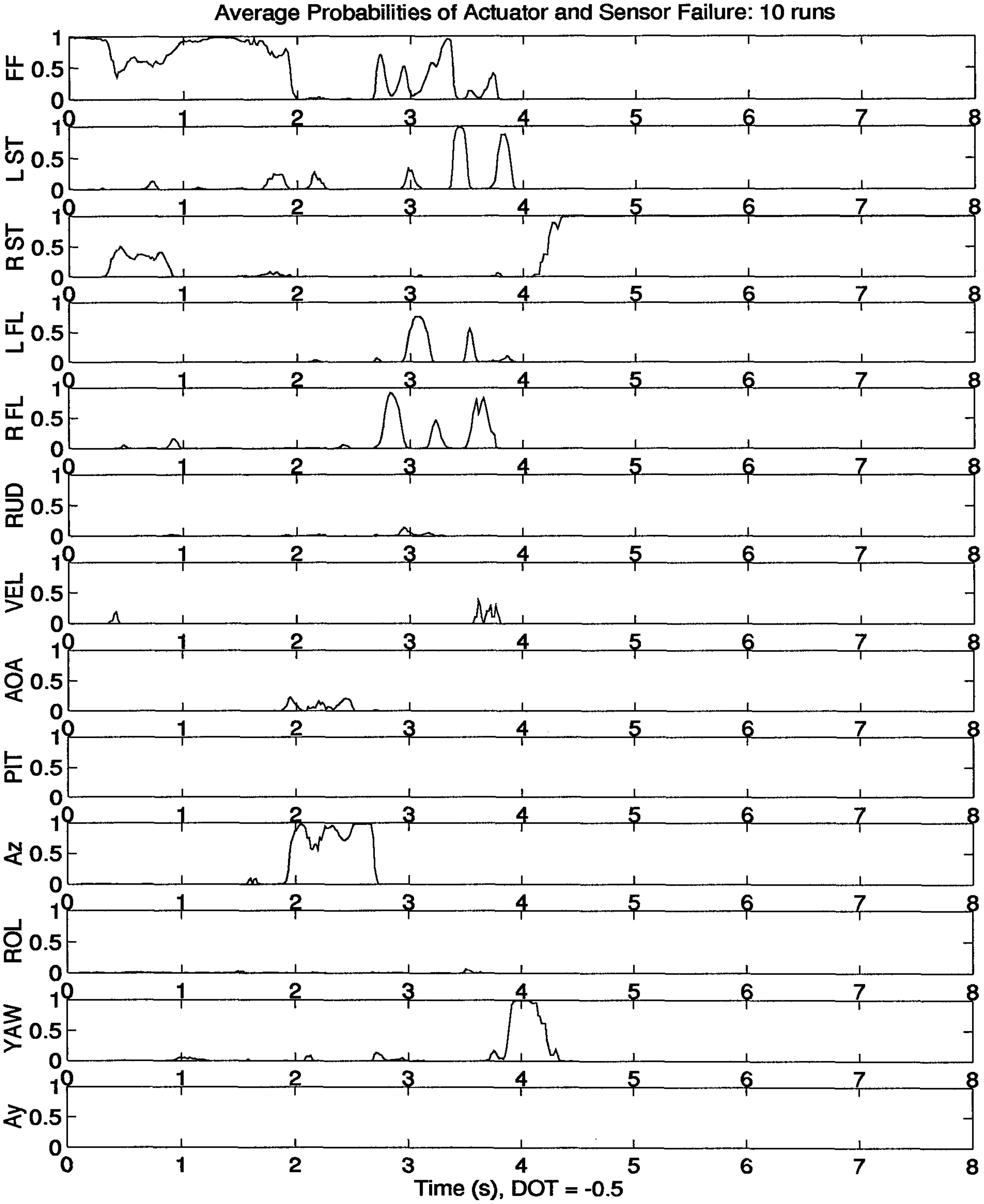
Notes: Filters = Q : Menke Dither : Actuator Break = 14

Figure 10. Fully Functional Aircraft with Original Q



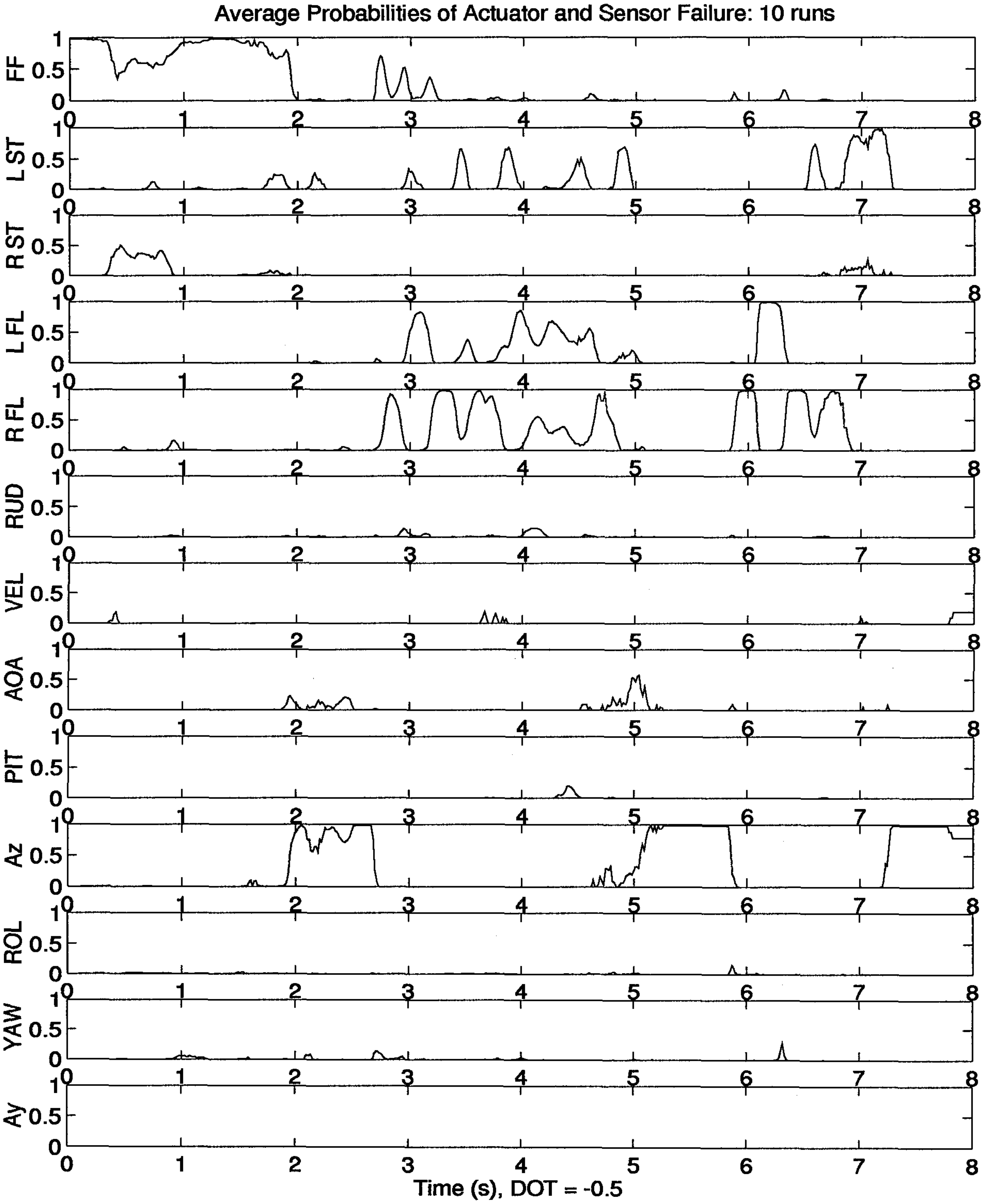
Notes: Filters = Q : Menke Dither : Actuator Break = 14

Figure 11. Simulated *Left Stabilator* Actuator Failure with Original Q



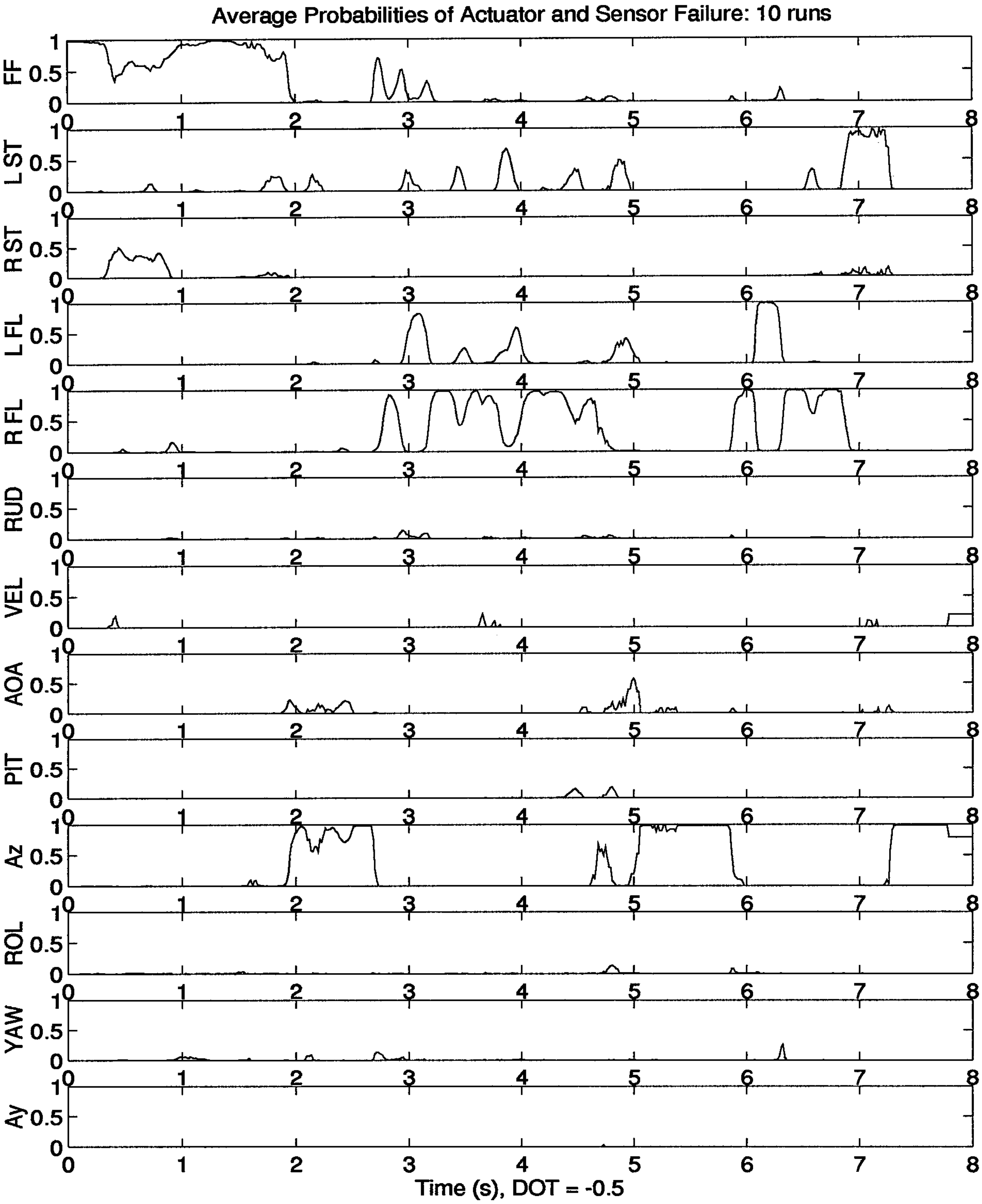
Notes: Filters = Q : Menke Dither : Actuator Break = 14

Figure 12. Simulated *Right Stabilator* Actuator Failure with Original Q



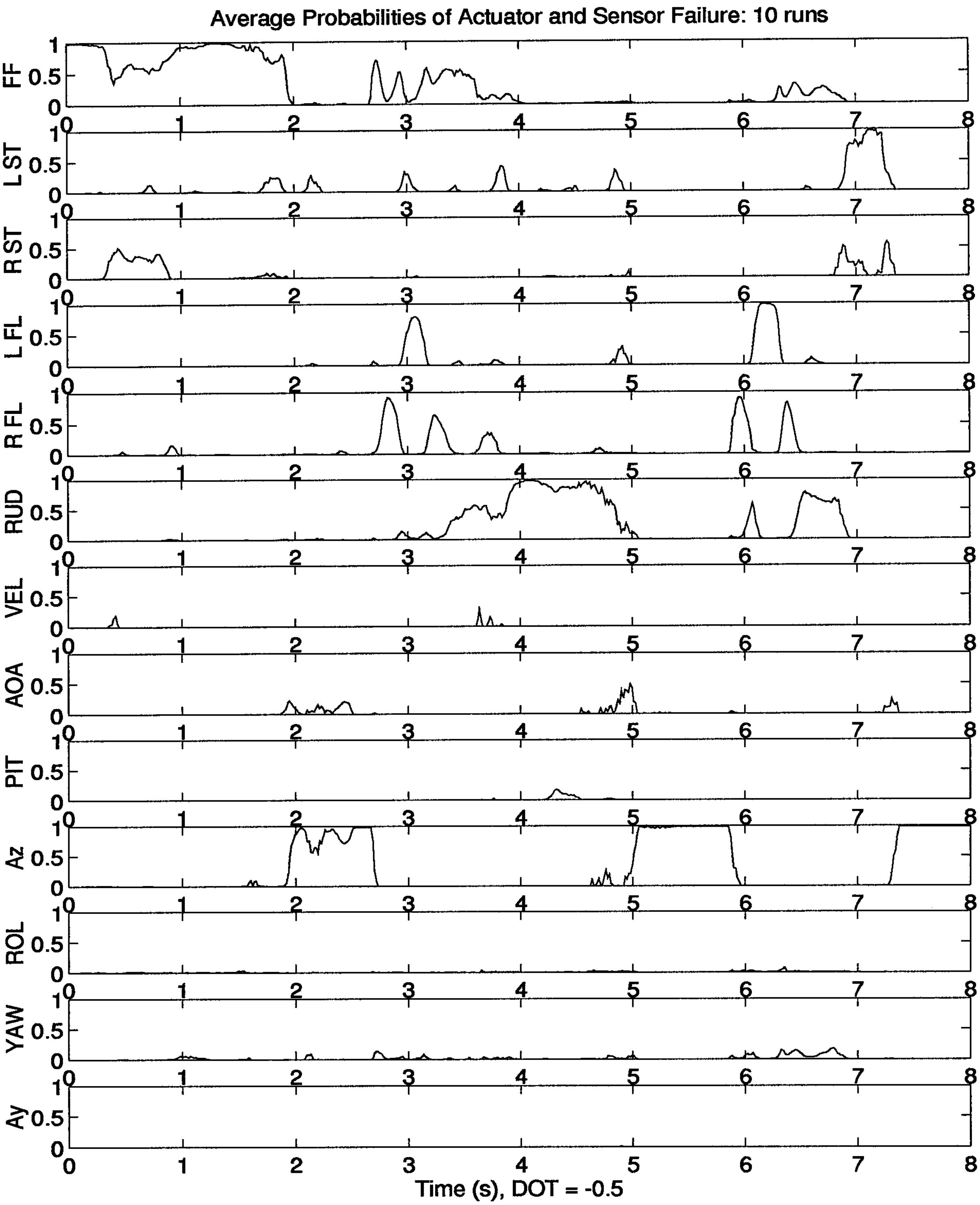
Notes: Filters = Q : Menke Dither : Actuator Break = 14

Figure 13. Simulated *Left Flaperon* Actuator Failure with Original Q



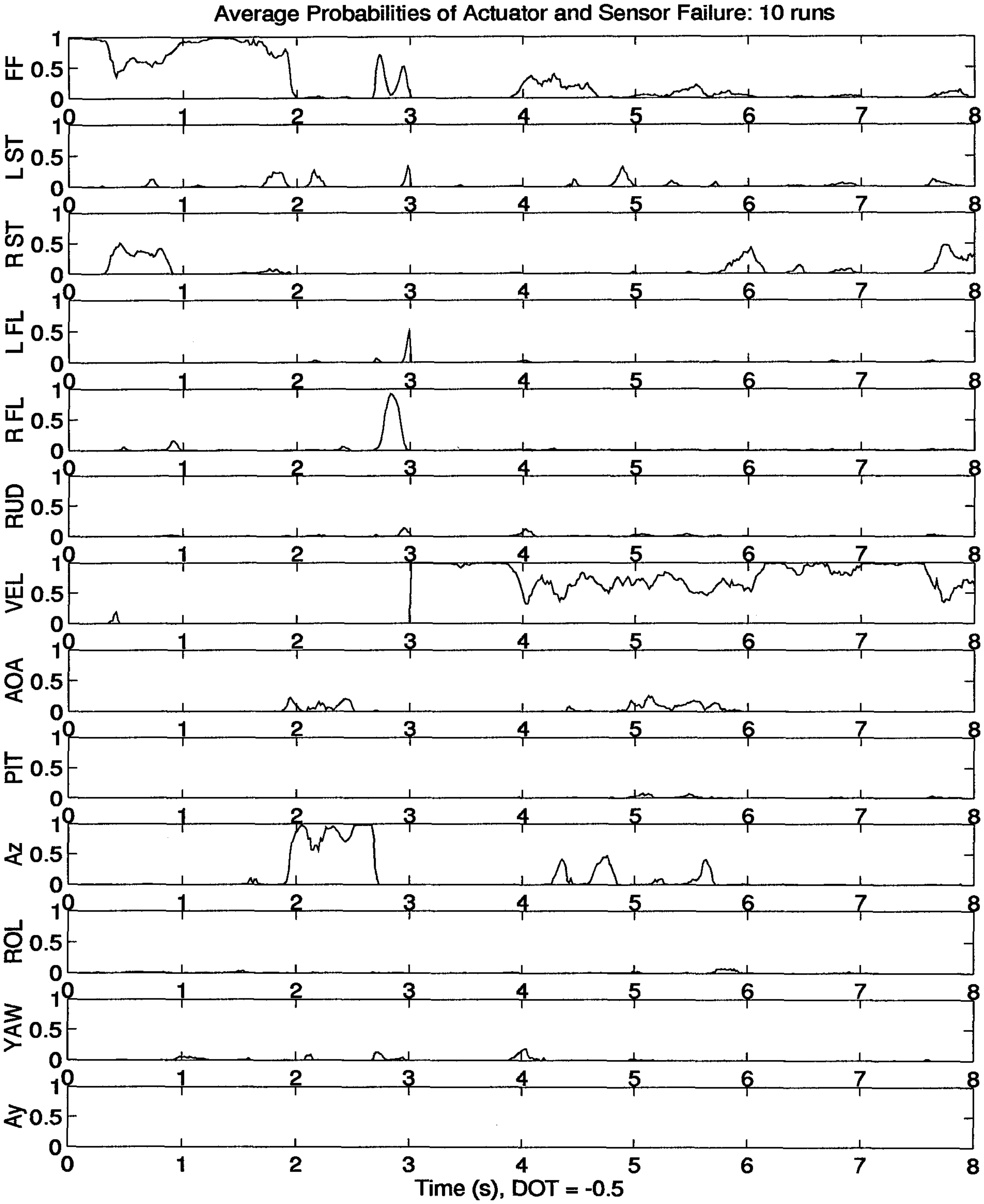
Notes: Filters = Q : Menke Dither : Actuator Break = 14

Figure 14. Simulated *Right Flaperon* Actuator Failure with Original Q



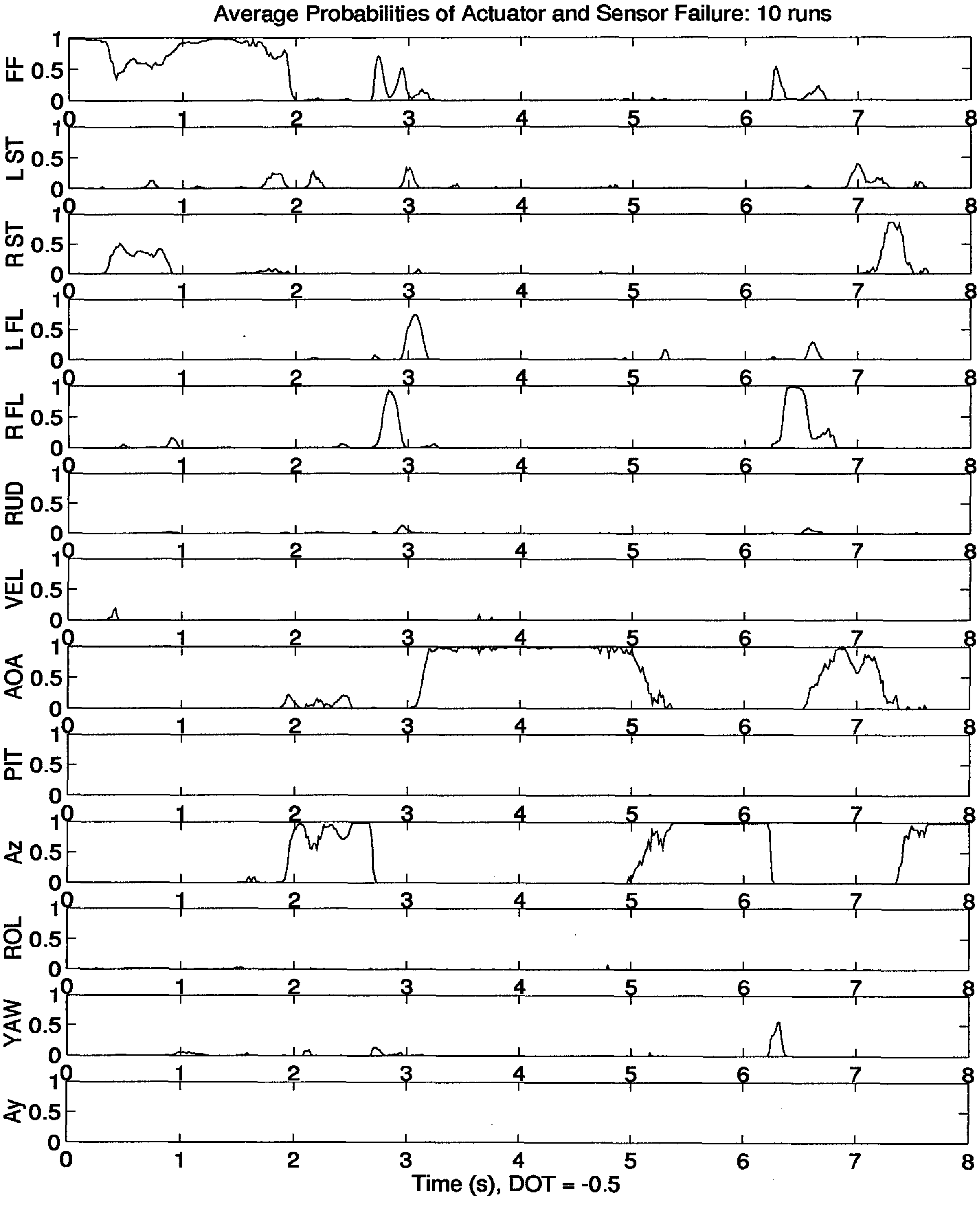
Notes: Filters = Q : Menke Dither : Actuator Break = 14

Figure 15. Simulated *Rudder* Actuator Failure with Original **Q**



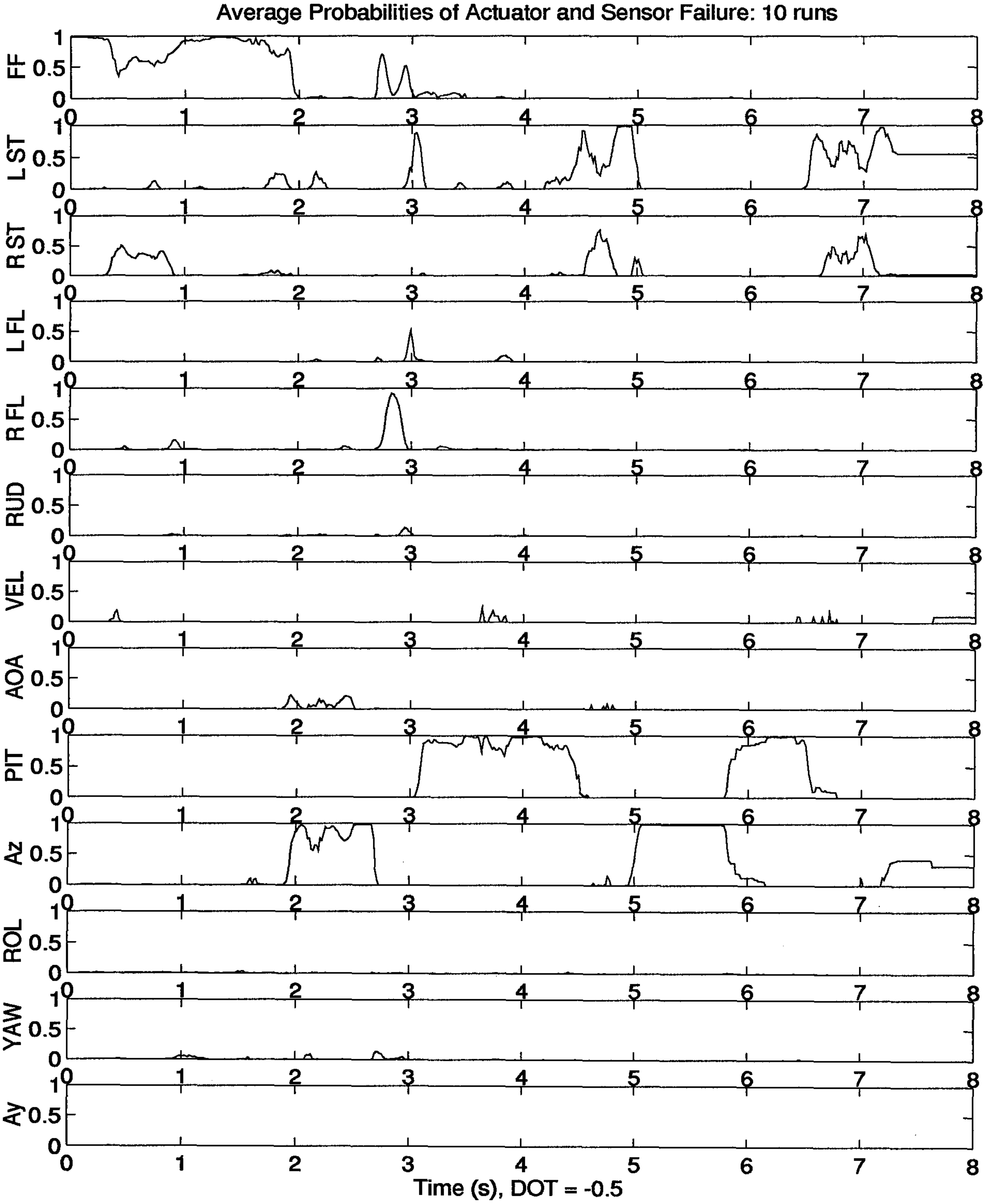
Notes: Filters = Q : Menke Dither : Actuator Break = 14

Figure 16. Simulated *Forward Velocity* Sensor Failure with Original Q



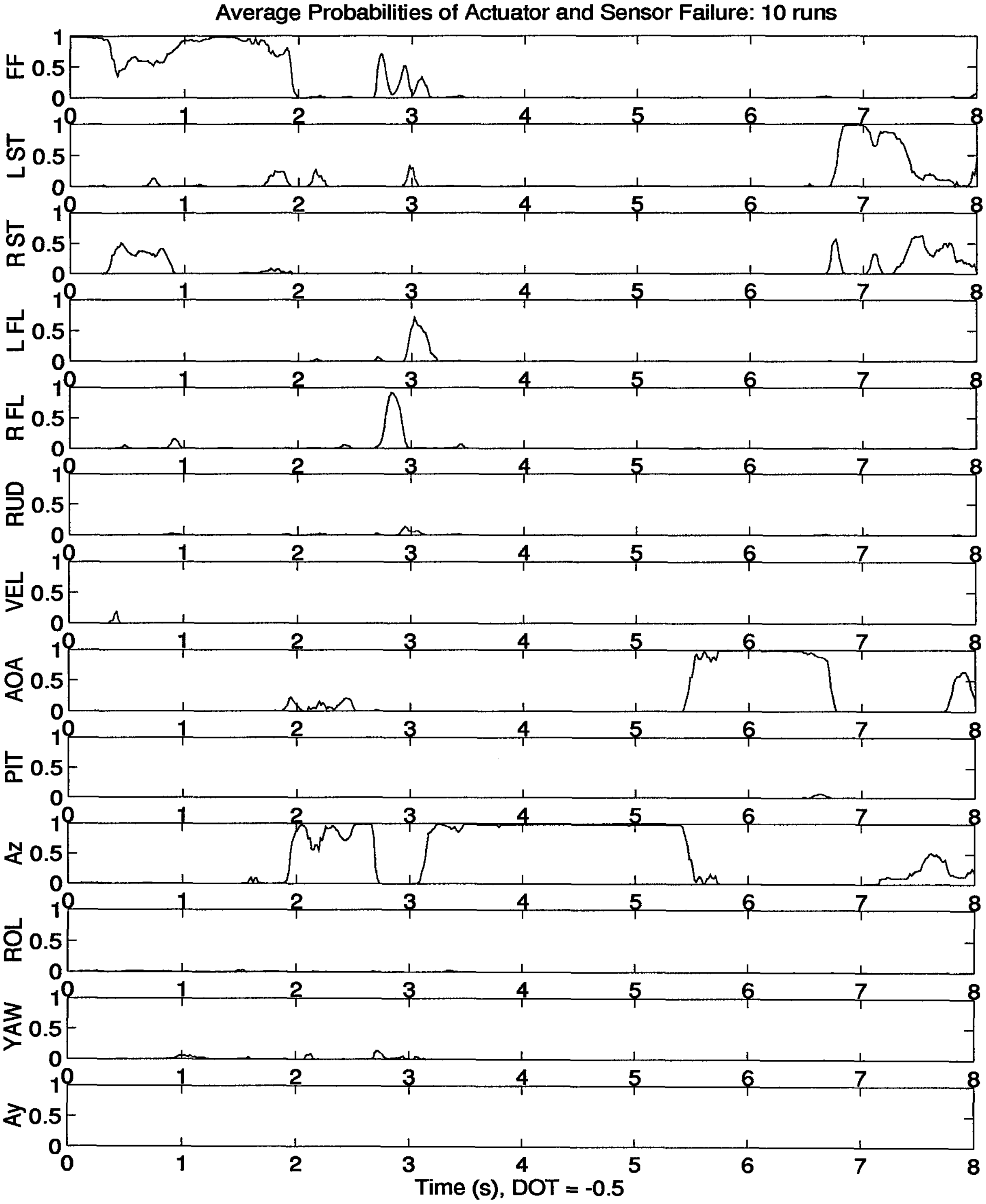
Notes: Filters = Q : Menke Dither : Actuator Break = 14

Figure 17. Simulated *Angle of Attack* Sensor Failure with Original Q



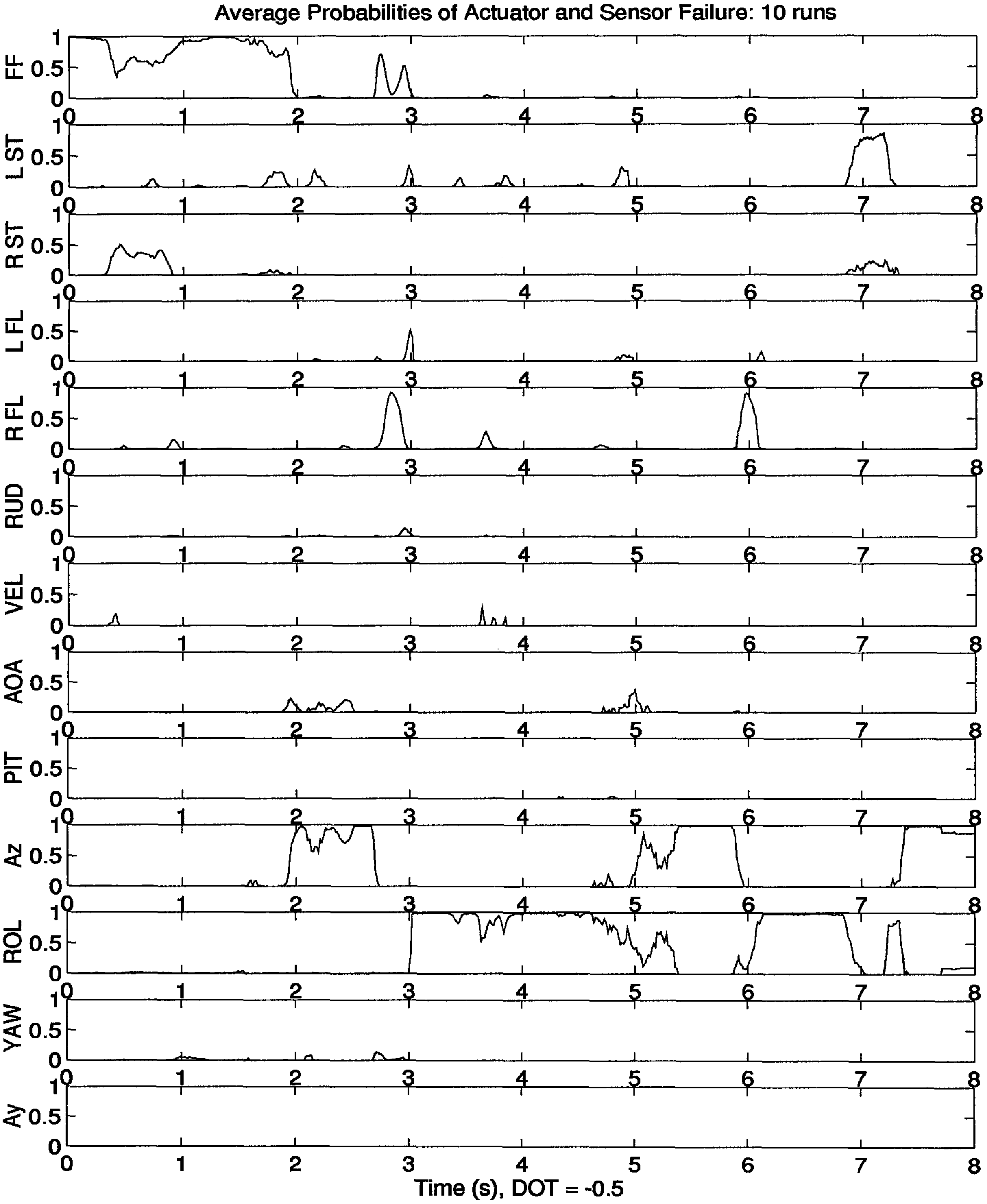
Notes: Filters = Q : Menke Dither : Actuator Break = 14

Figure 18. Simulated *Pitch Rate* Sensor Failure with Original **Q**



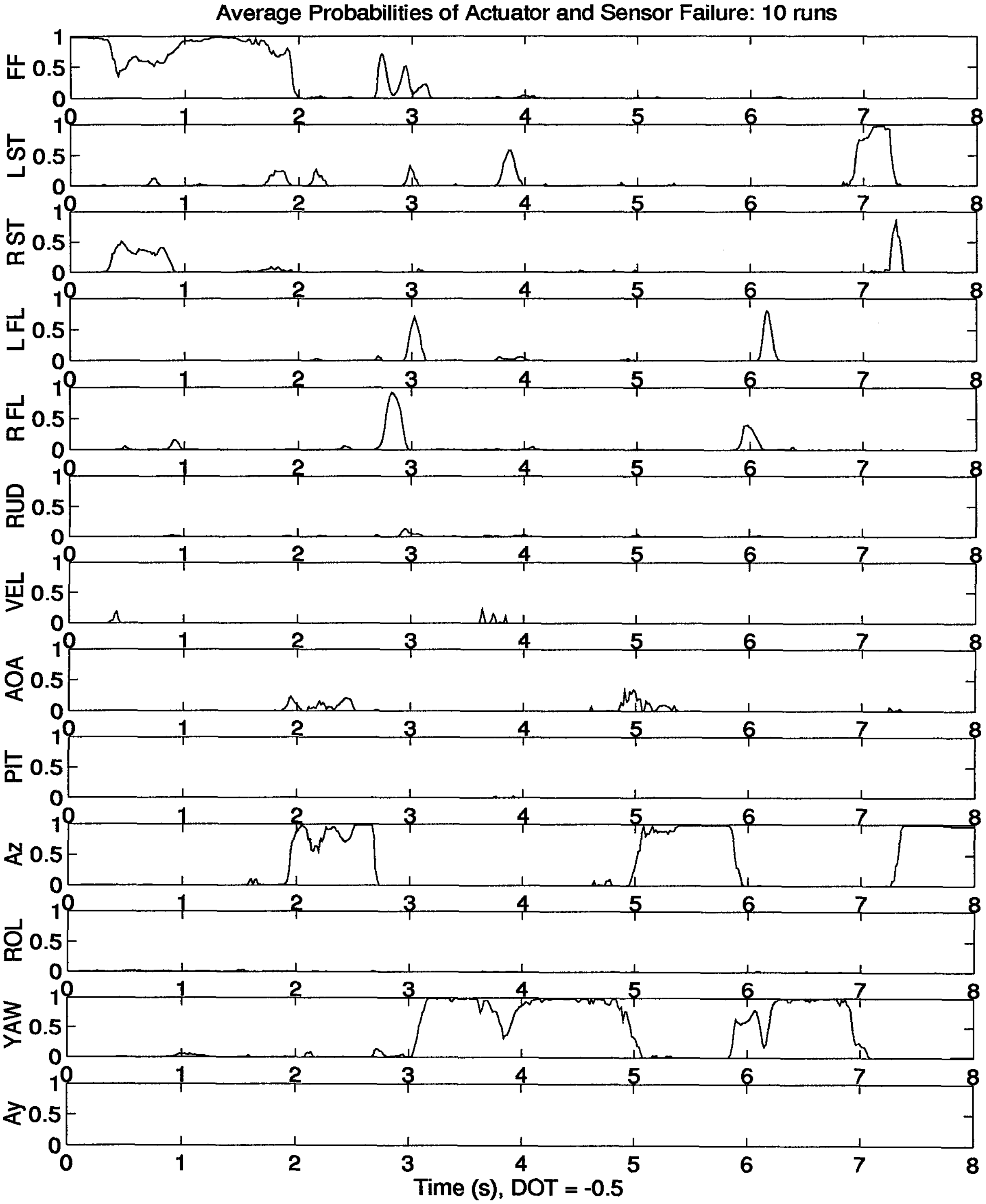
Notes: Filters = Q : Menke Dither : Actuator Break = 14

Figure 19. Simulated *Normal Acceleration* Sensor Failure with Original Q



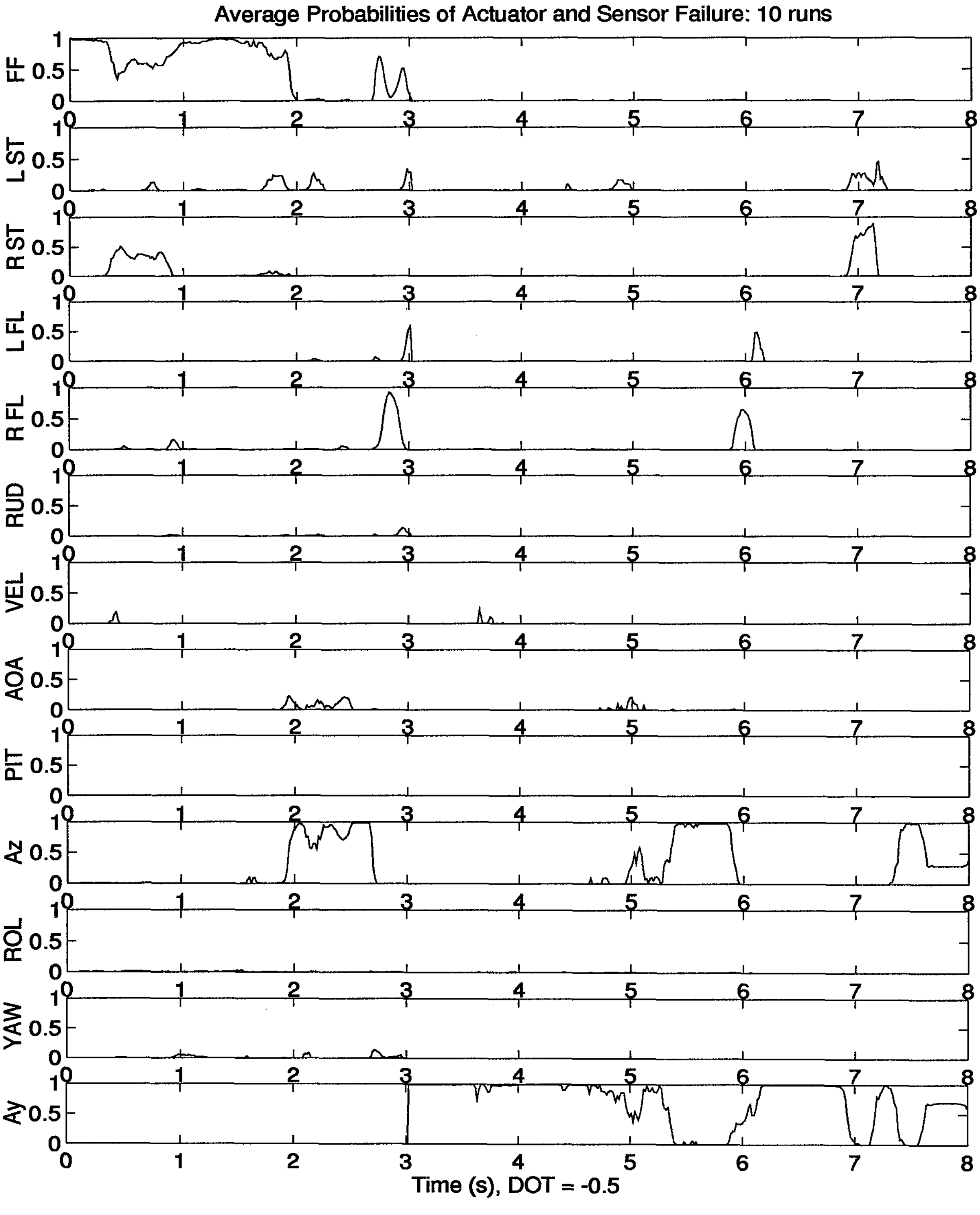
Notes: Filters = Q : Menke Dither : Actuator Break = 14

Figure 20. Simulated *Roll Rate* Sensor Failure with Original Q



Notes: Filters = Q : Menke Dither : Actuator Break = 14

Figure 21. Simulated *Yaw Rate* Sensor Failure with Original Q



Notes: Filters = Q : Menke Dither : Actuator Break = 14

Figure 22. Simulated *Lateral Acceleration* Sensor Failure with Original **Q**

4.2.1.2 Techniques for Performance Improvement. There are a number of MMAE performance improvement techniques available. Chapter 2 discussed a few, of which some are now standard and others applicable depending on the circumstances. The key is to recognize that most are really application dependent. What may work well in one implementation may not be a panacea for all. In that light, this effort worked on some different, more direct techniques to improve performance. The most effective technique admitted, and attempted to correct for, uncertainty in our design model through direct addition of pseudonoise in the propagation equations. Also effective, though somewhat less so, is the addition of pseudonoise in the measurement models for the update equations. Yet a third technique, ineffective for our purposes but noteworthy for future applications, attempts to account for uncertainty resultant from a troublesome actuator. Finally, we discuss effecting performance improvement through dither modification. The following subsections describe a progression of these techniques and their associated tradeoffs.

Direct Pseudonoise on Longitudinal Q_d Entries: One way in which to account for uncertainty within a propagation model is simply to add pseudonoise. Of course there is pseudonoise already present within the framework of the Dryden wind model. One approach might seek to add the pseudonoise by increasing the **Q** entries associated with the Dryden model. Unfortunately, the manner in which uncertainty enters the equations in the Dryden model is strictly governed by **G** and $\mathbf{\Phi}$ as shown below

$$\mathbf{Q}_d = \int_{t_{i-1}}^{t_i} \mathbf{\Phi G Q G^T \Phi^T} dt \tag{56}$$

If a designer were to add more noise by increasing **Q** above, the effect on performance would be difficult to predict, and there would actually be limits to what could be accomplished channel-by-channel, as a result of the coupling within **G** and $\mathbf{\Phi}$. A more direct approach is to add a diagonal matrix of pseudonoise at the $\mathbf{Q}_d$ level. This is equivalent to adding a second source of discrete noise as shown in the equation below.

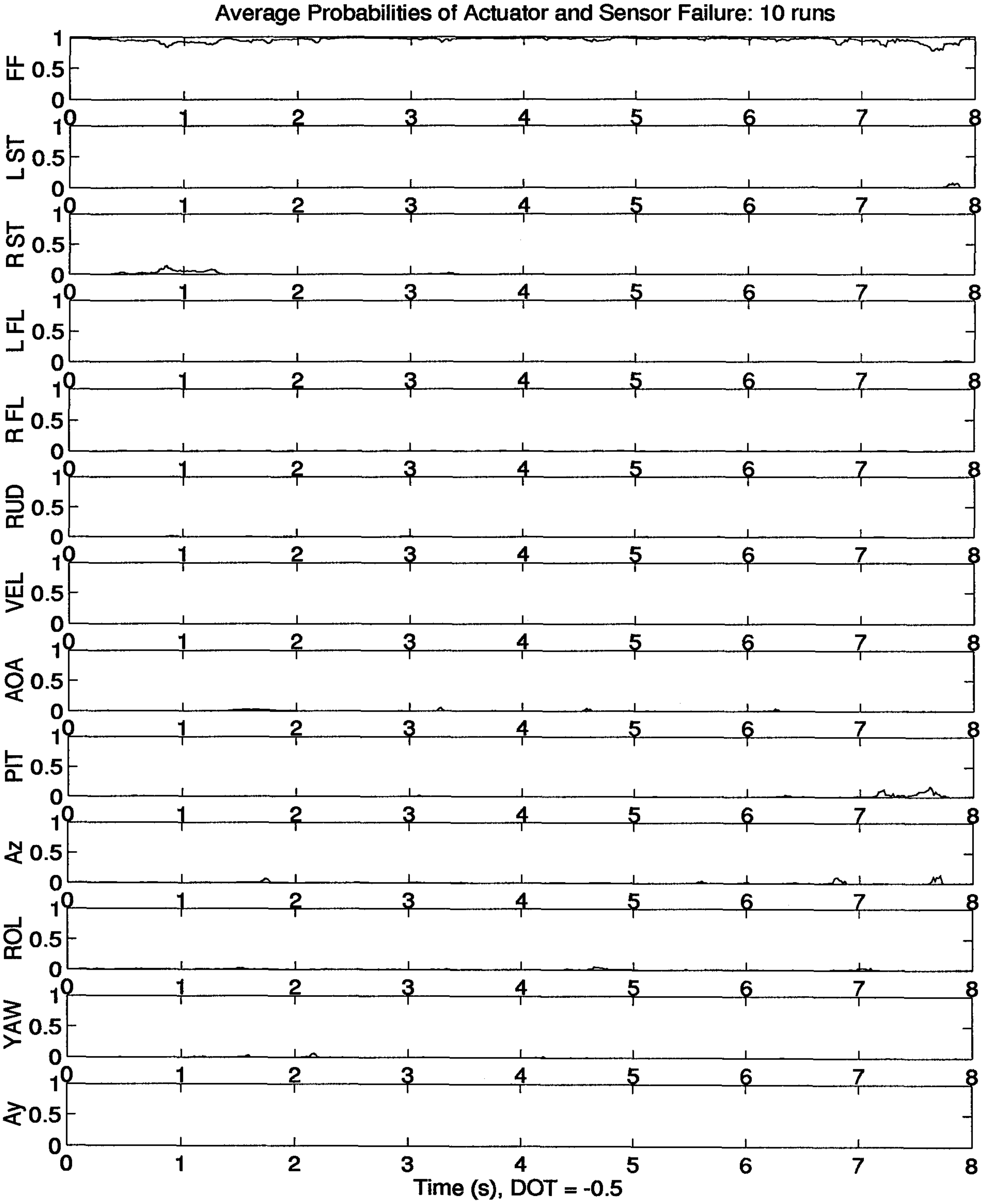
$$\mathbf{x}_k(t_{i+1}) = \mathbf{\Phi}_k \mathbf{x}_k(t_i) + \mathbf{B}_{\mathbf{d}k}\mathbf{u}(t_i) + \mathbf{G}_{\mathbf{d}k}\mathbf{w}_{\mathbf{d}k}(t_i) + \mathbf{w}'_{\mathbf{d}k}(t_i) \tag{57}$$

In effect we are saying that there is a source of uncertainty unknown to us, but whose effect produces a diagonal $\mathbf{Q}_d$ type matrix. Within such a matrix, we are free to choose the diagonal entries arbitrarily and thus affect the amount of uncertainty within each channel directly.

Before proceeding to the results and tradeoffs, we must yet acknowledge that blind adjustment of $\mathbf{Q}_d$, even when *assumed* to be diagonal (i.e. adjustment of eight numbers in this case), is still a formidable task. We therefore looked to the results of the untuned algorithm for clues as to which channels may or may not require tuning. One false alarm which repeatedly appears in Figures 10 to 22 is the normal acceleration sensor. From this we can narrow our attention for now, to the longitudinal channel.

Figures 23 through 35 demonstrate the performance improvement possible as a result of direct addition of pseudonoise to the longitudinal channel entries of $\mathbf{Q}_d$. In fact, the values used to arrive at this performance are shown below:

$$\mathbf{Q}_{d_{long}} = \begin{bmatrix} 1\times10^{-5} & 0 & 0 & 0 \\ 0 & 2\times10^{-2} & 0 & 0 \\ 0 & 0 & 1\times10^{-7} & 0 \\ 0 & 0 & 0 & 1\times10^{-8} \end{bmatrix} \tag{58}$$



Notes: Filters = Q + long pseudonoise: Menke Dither : Actuator Break = 14

Figure 23. Fully Functional Aircraft with Longitudinal Pseudonoise

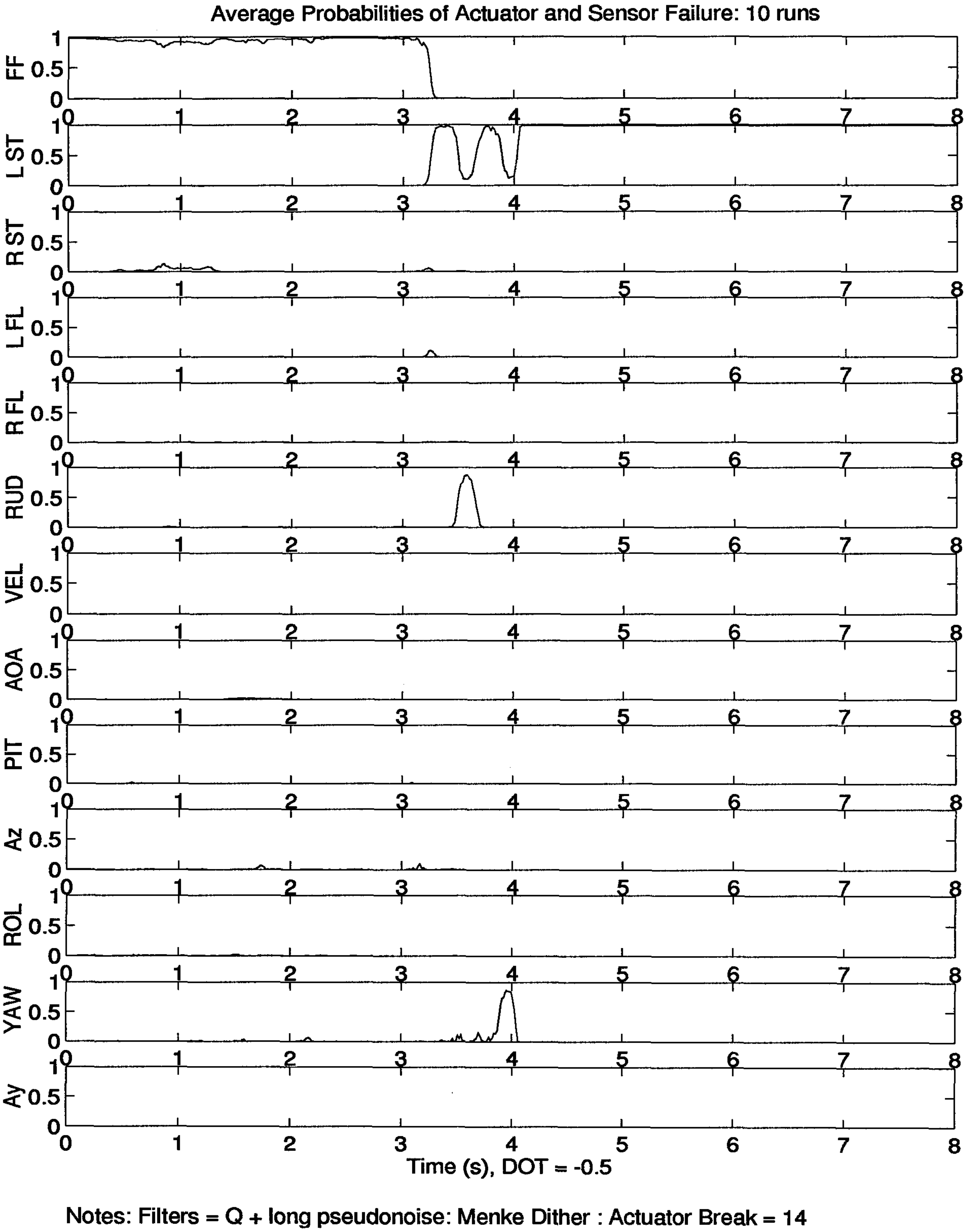
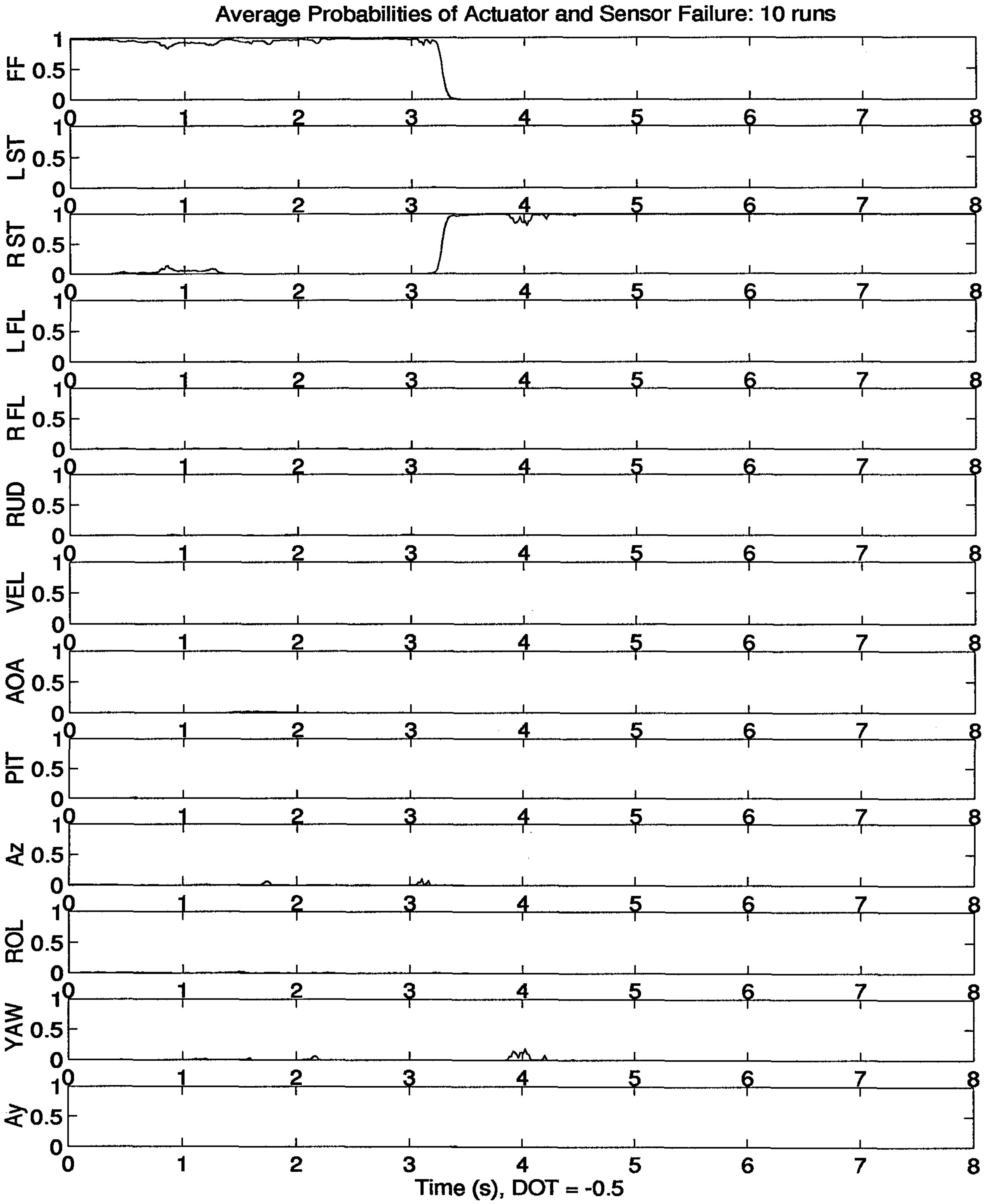
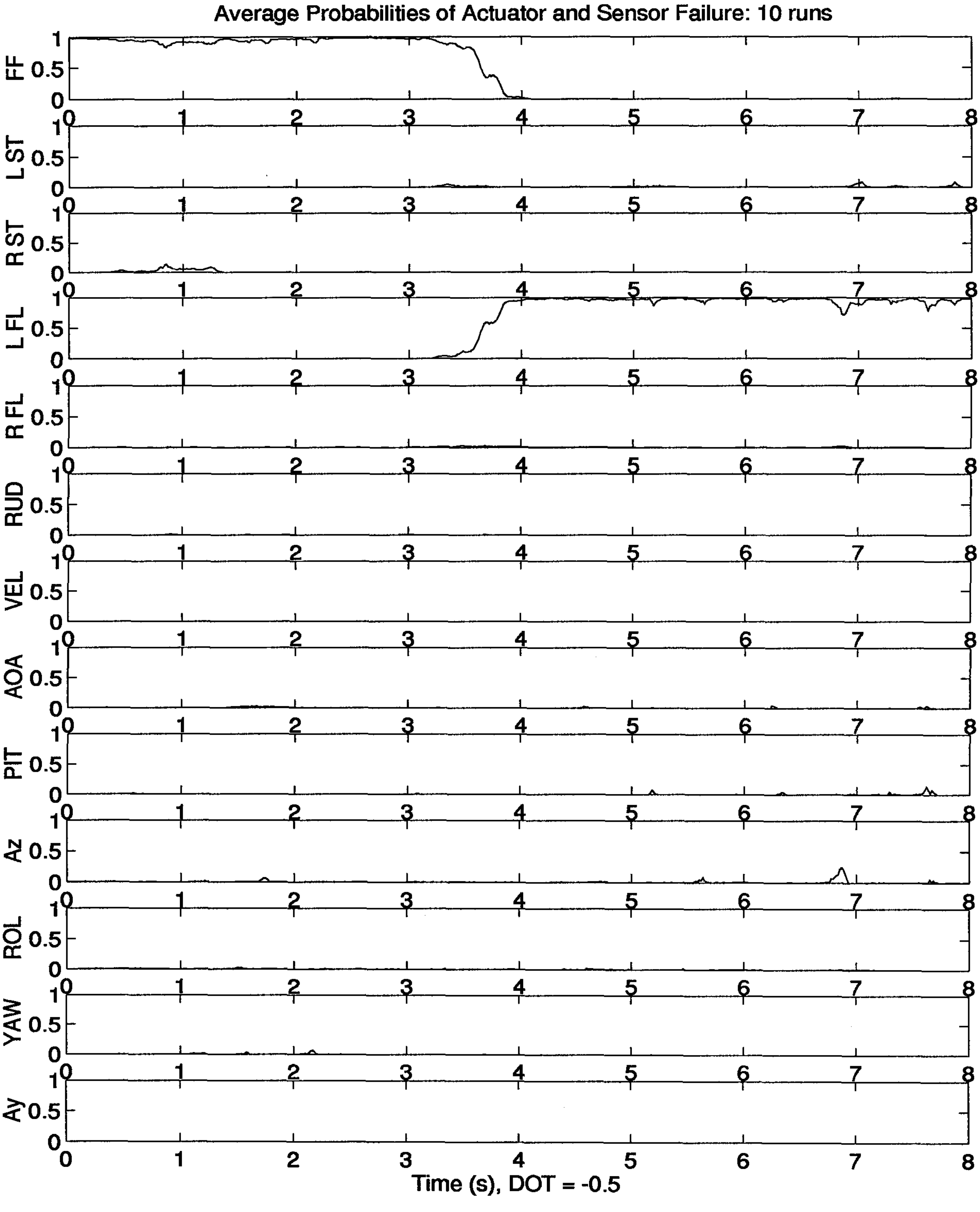


Figure 24. Simulated *Left Stabilator* Actuator Failure with Longitudinal Pseudonoise



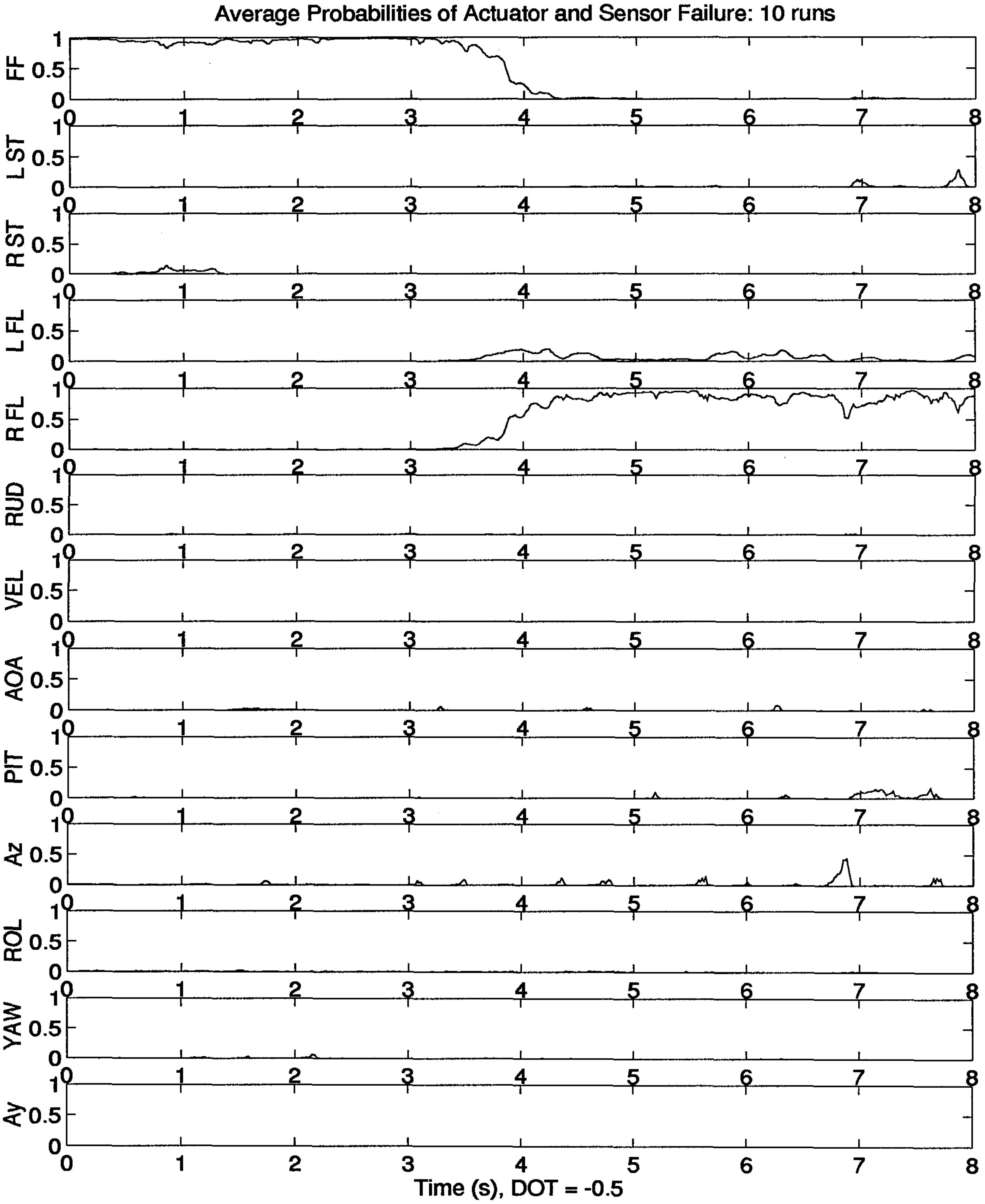
Notes: Filters = Q + long pseudonoise: Menke Dither : Actuator Break = 14

Figure 25. Simulated *Right Stabilator* Actuator Failure with Longitudinal Pseudonoise



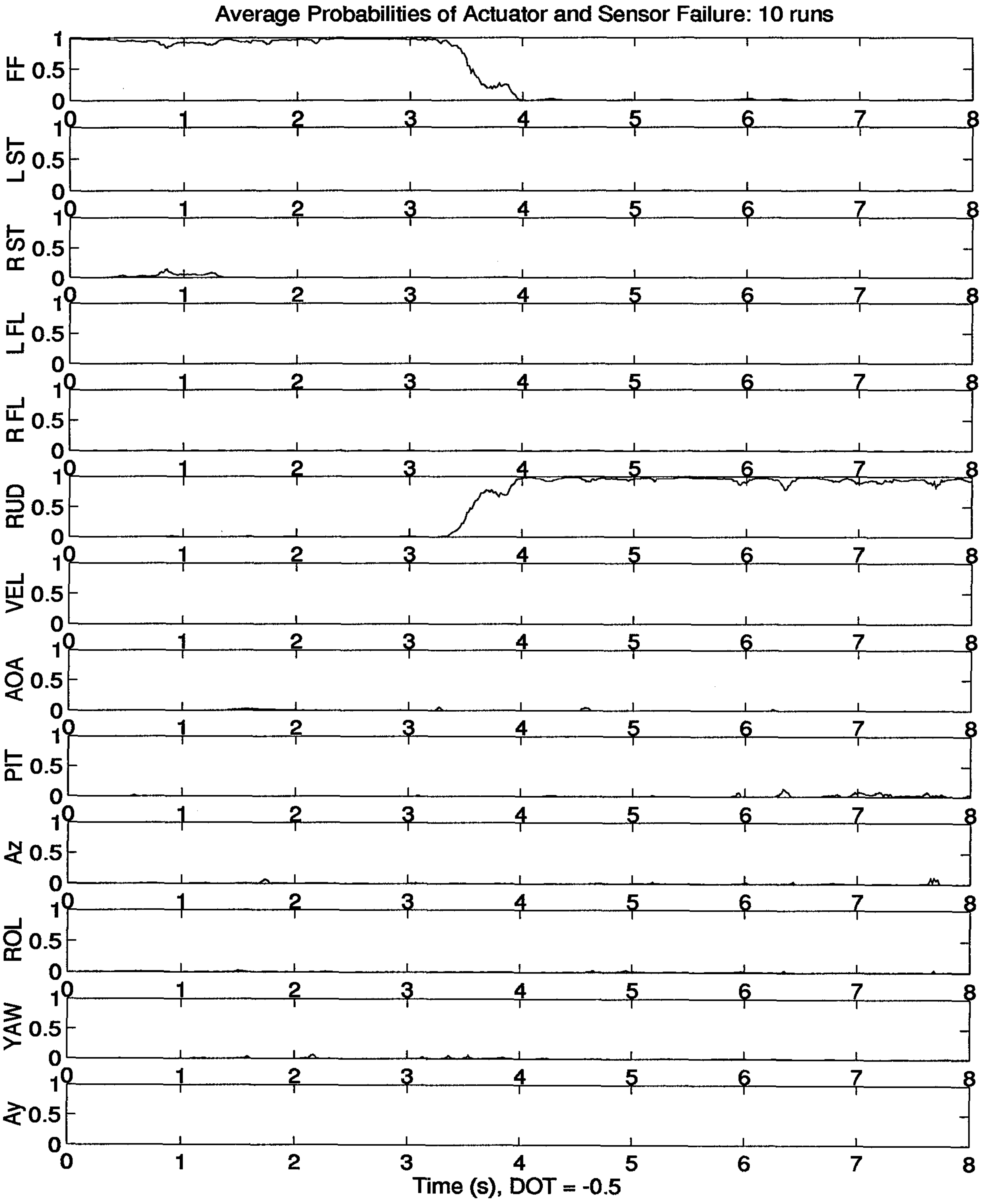
Notes: Filters = Q + long pseudonoise: Menke Dither : Actuator Break = 14

Figure 26. Simulated *Left Flaperon* Actuator Failure with Longitudinal Pseudonoise



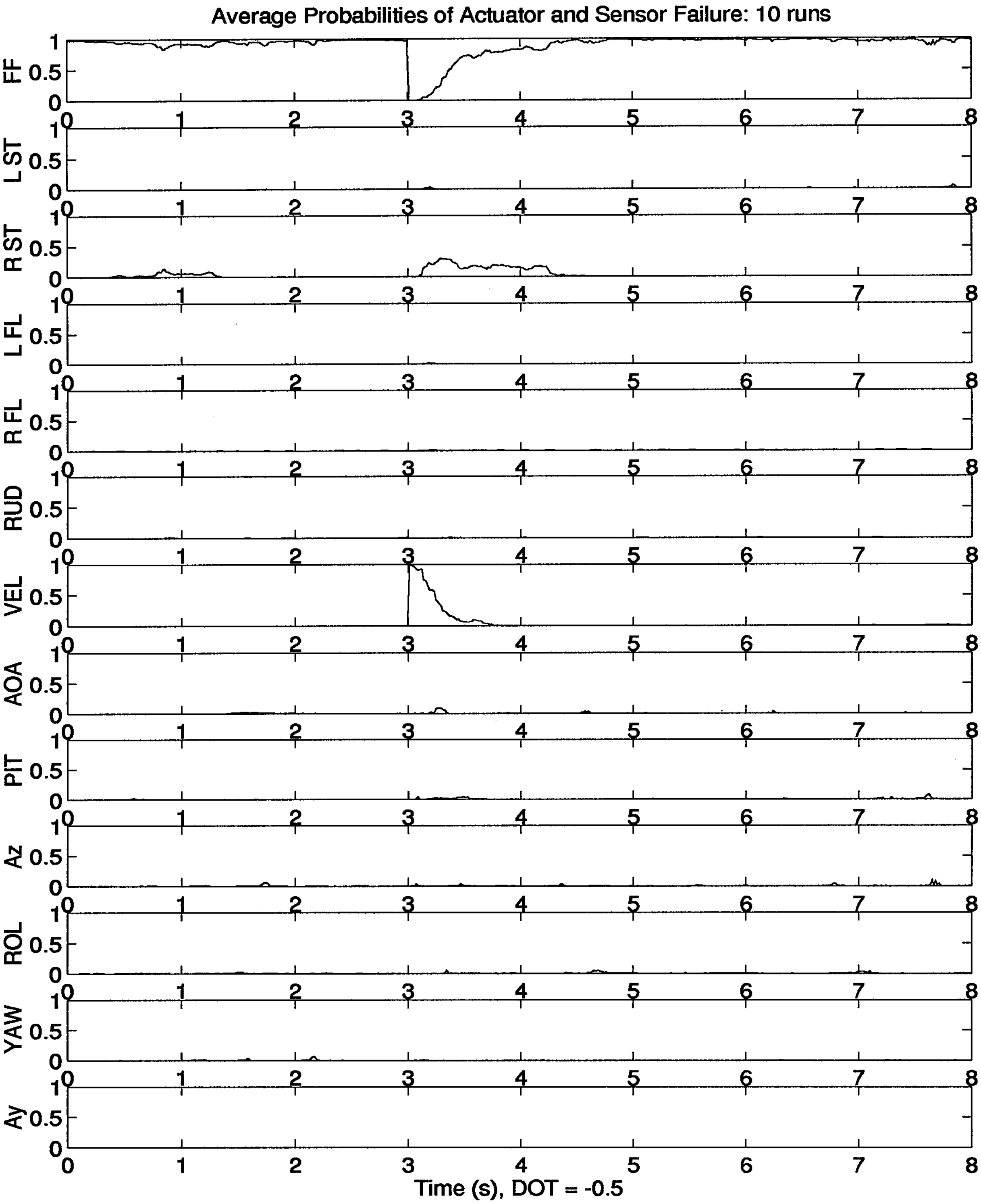
Notes: Filters = Q + long pseudonoise: Menke Dither : Actuator Break = 14

Figure 27. Simulated *Right Flaperon* Actuator Failure with Longitudinal Pseudonoise



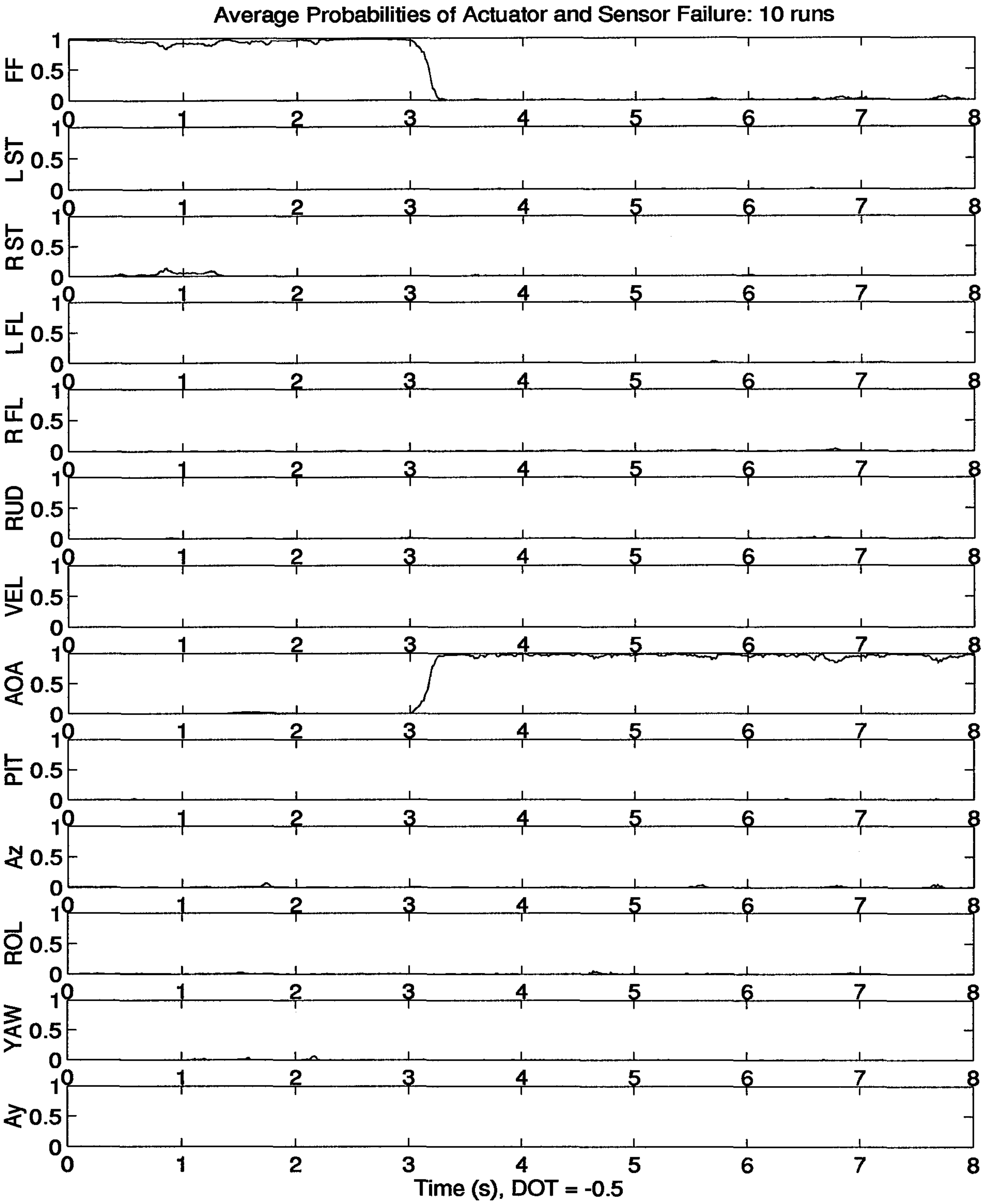
Notes: Filters = Q + long pseudonoise: Menke Dither : Actuator Break = 14

Figure 28. Simulated *Rudder* Actuator Failure with Longitudinal Pseudonoise



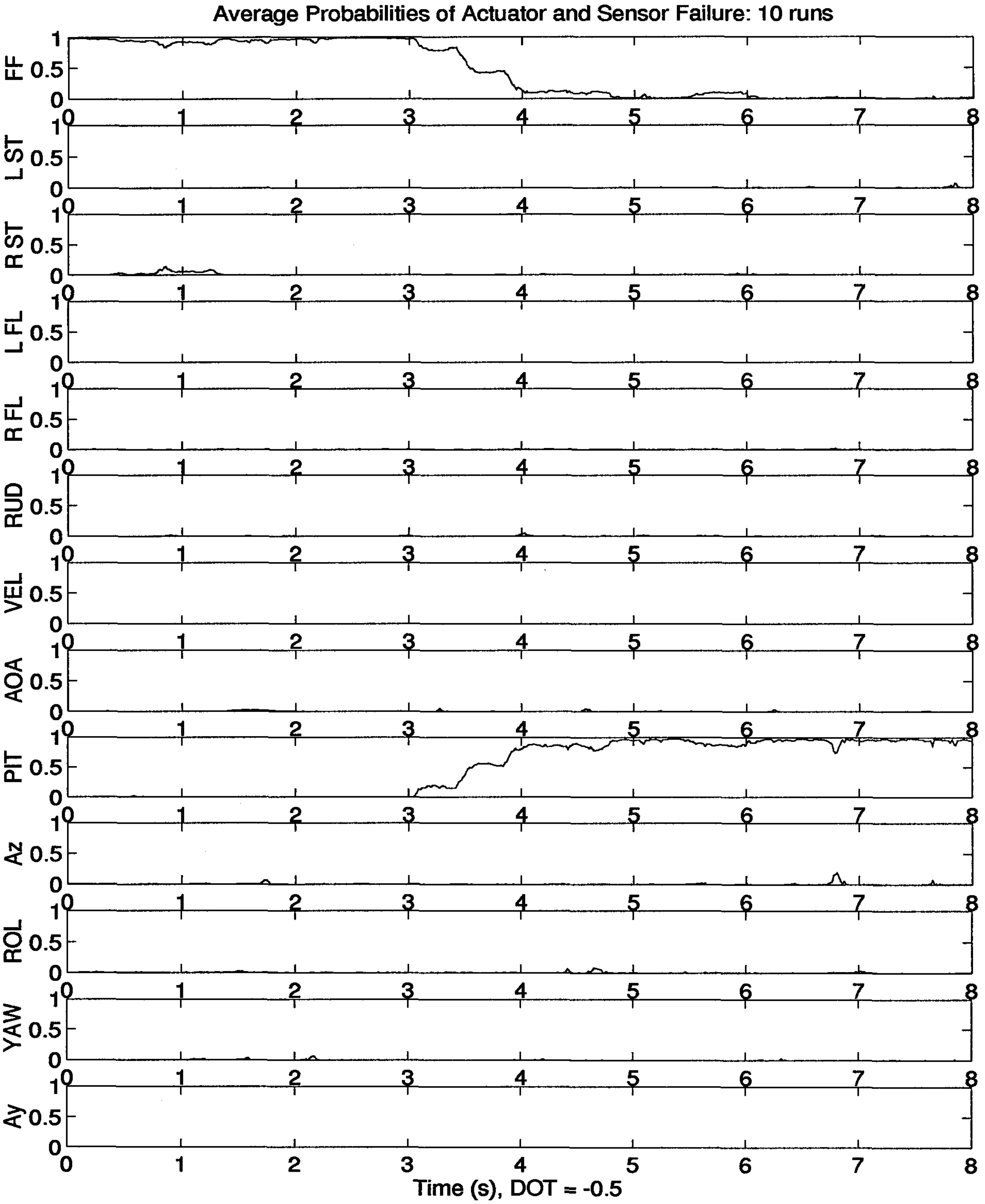
Notes: Filters = Q + long pseudonoise: Menke Dither : Actuator Break = 14

Figure 29. Simulated *Forward Velocity* Sensor Failure with Longitudinal Pseudonoise



Notes: Filters = Q + long pseudonoise: Menke Dither : Actuator Break = 14

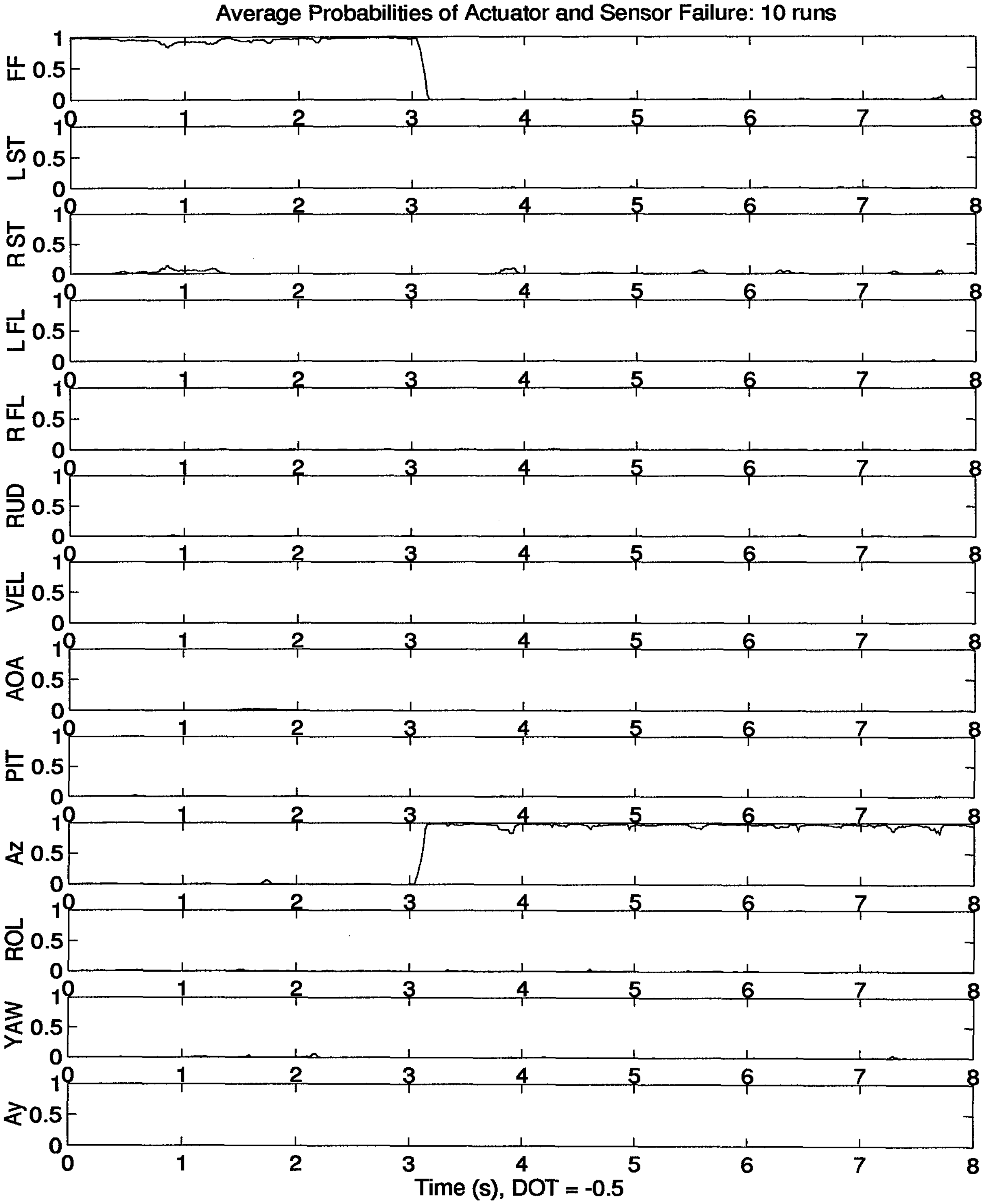
Figure 30. Simulated *Angle of Attack* Sensor Failure with Longitudinal Pseudonoise



Average Probabilities of Actuator and Sensor Failure: 10 runs

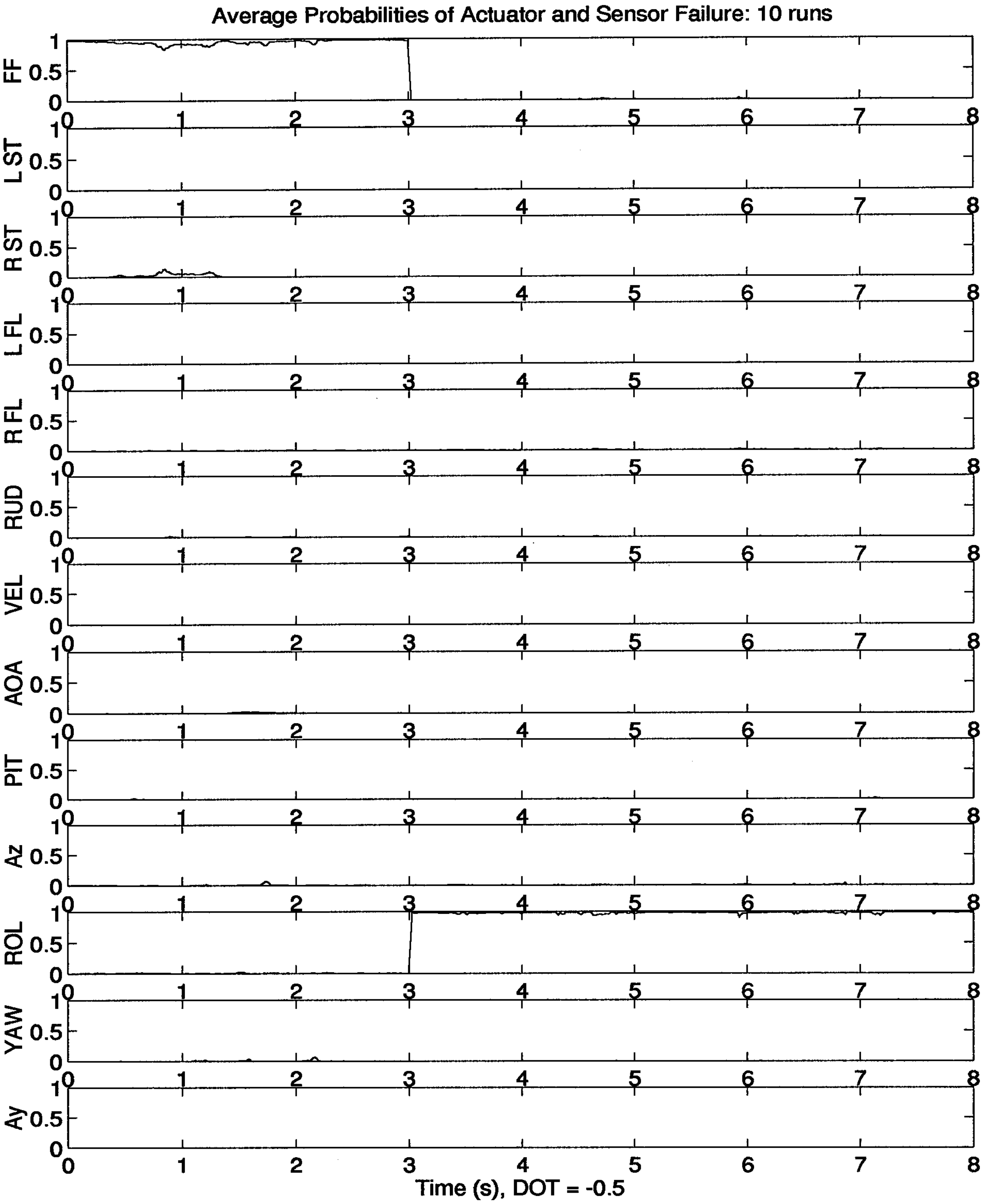
Notes: Filters = Q + long pseudonoise: Menke Dither : Actuator Break = 14

Figure 31. Simulated *Pitch Rate* Sensor Failure with Longitudinal Pseudonoise



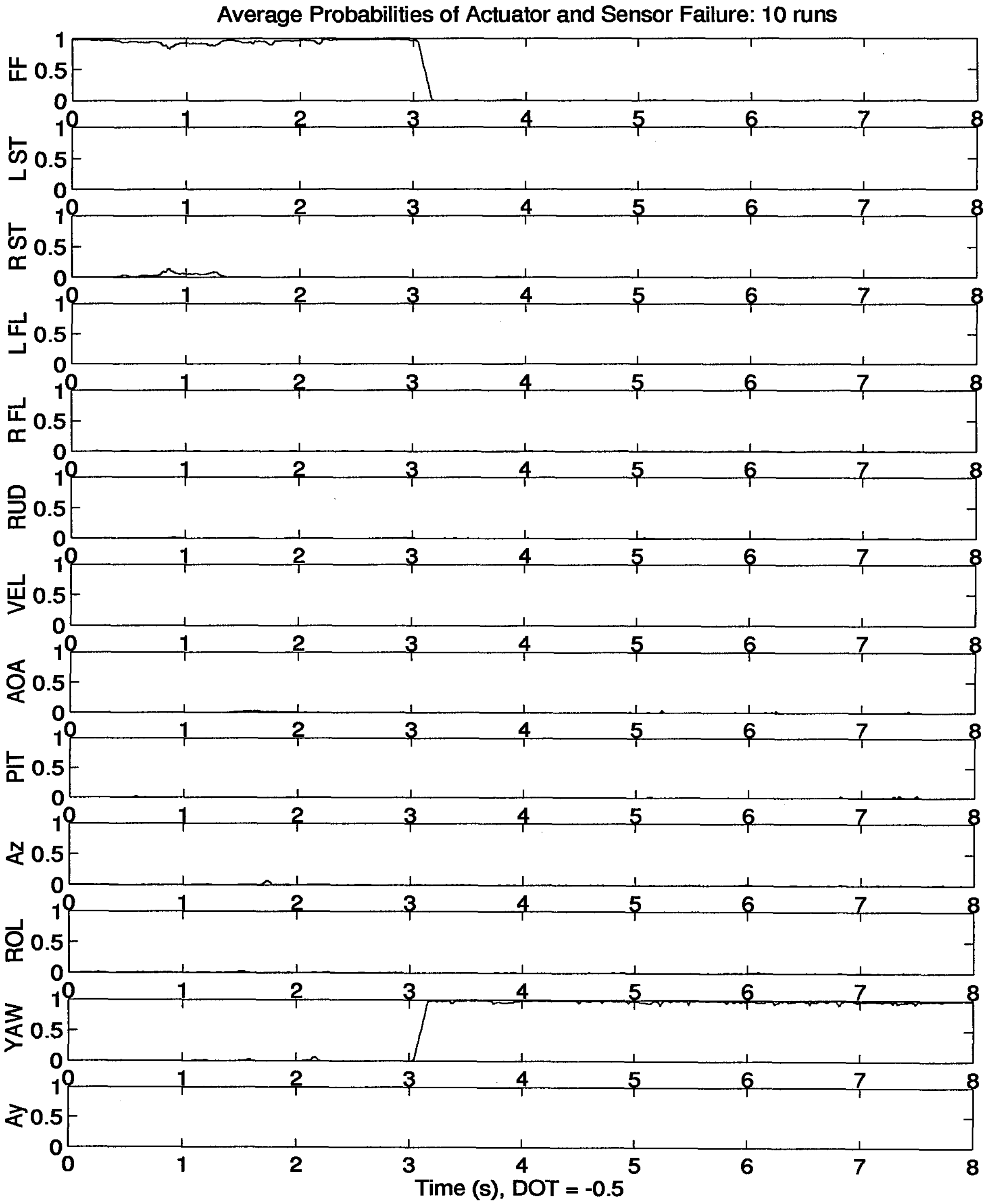
Notes: Filters = Q + long pseudonoise: Menke Dither : Actuator Break = 14

Figure 32. Simulated *Normal Acceleration* Sensor Failure with Longitudinal Pseudonoise



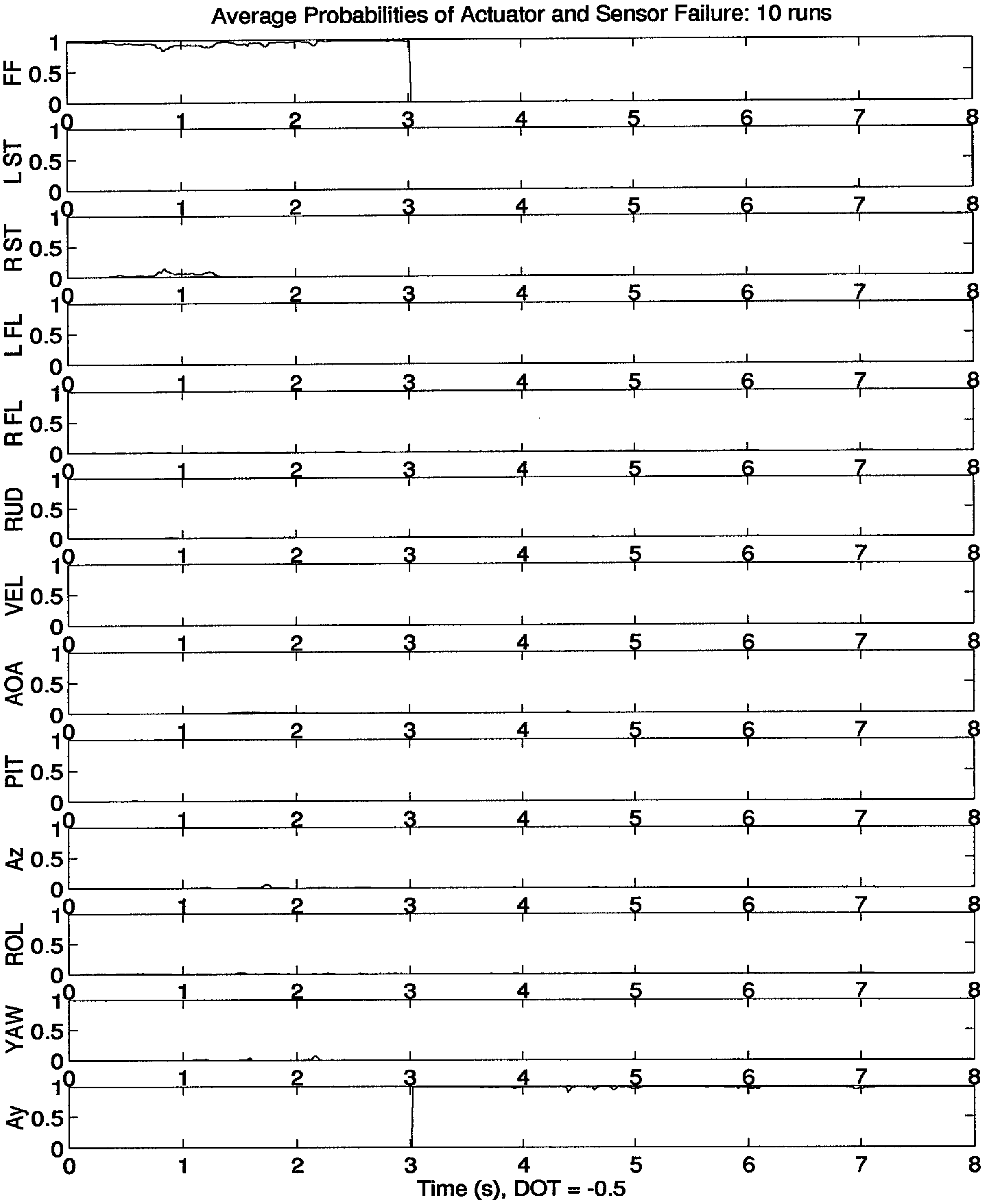
Notes: Filters = Q + long pseudonoise: Menke Dither : Actuator Break = 14

Figure 33. Simulated *Roll Rate* Sensor Failure with Longitudinal Pseudonoise



Notes: Filters = Q + long pseudonoise: Menke Dither : Actuator Break = 14

Figure 34. Simulated *Yaw Rate* Sensor Failure with Longitudinal Pseudonoise



Notes: Filters = Q + long pseudonoise: Menke Dither : Actuator Break = 14

Figure 35. Simulated *Lateral Acceleration* Sensor Failure with Longitudinal Pseudonoise

Now that we've achieved some level of acceptable performance, it becomes meaningful to take a close look. The thirteen failure conditions can be broken into a couple of smaller groups. Five actuator failures, **L ST**, **R ST**, **L FL**, **R FL**, and **RUD**, comprise one group. They have historically been the toughest to detect because there are no direct measurements of actuator position; also it takes time for the impact of an actuator failure to show up in sensor outputs. Sensor failures are broken down into a longitudinal group, **VEL**, **AOA**, **PIT**, and **Az**, and a lateral group, **ROL**, **YAW**, and **Ay**. This is natural as a result of the decoupling which exists between each channel. The last group is the lone no-failure condition, **FF**. We shall address these latest results by subgroup.

The fully functional case results shown in Figure 23 displays substantial performance improvement over the untuned case, Figure 10. Visible at the beginning of the run is a slight dip in probability, easily attributable to transient or settling behavior. Somewhat more problematic though, is the dip which occurs near the end of the run. A good amount of probability is spilling into the failed pitch rate hypothesis. Actually this is the result of a tradeoff between false alarms and sensitivity. It is quite possible to eliminate this dip altogether by increasing the $\mathbf{Q}_d$ pseudonoise entry associated with pitch rate. Unfortunately the algorithm then becomes insensitive to an actual failure on the pitch rate sensor and will miss it during normal operation. In fact, once we began to add pseudonoise to the longitudinal channel, we also began to degrade performance in other areas. An eyeball comparison between like failures in Figures 10 to 22 and Figures 23 to 35 bears this out. In the untuned case, indication of failure was often quite fast, but unsustained. Through tuning we've given up speed for stability. The effect is also unique for this trim and dither. As will be discussed later, modification of the dither may remove the need for this tradeoff.

The actuators also continue to demonstrate improved failure declaration performance brought about through direct addition of pseudonoise to the longitudinal entries of $\mathbf{Q}_d$. Both trends discussed in the fully functional case continue to be a concern, but Figures 24 to 28 clearly show that in all cases, the proper channels are being excited. Figure 24 shows that, after a second or so of bouncing, the failed left stabilator hypothesis holds fast. For some applications, bouncing like this may be unacceptable. We will return to this issue later. With less ambiguity, the right stabilator failure declaration comes up quickly and holds fast in Figure 25. It first appears odd that similar probability bouncing isn't noted in both stabilators. This, though, can be attributed to secondary effects, like adverse yaw, which depend somewhat on the phasing of the control surface when the failure is introduced. It is quite possible that at 3.0 seconds, when the failure is a left stabilator, the position of the stabilators gives an initial yawing which phases in with the expected yaw and is interpreted as a yaw failure. The opposite yaw which takes place when the right stabilator is induced phases out of the concurrent yaw and does not push the failure indication over the threshold, so the false alarm is not declared. The flaperons posed a more difficult challenge. The right flaperon actuator failure probability was being erroneously attributed to the left flaperon. To correct this problem, we utilized a dither phase offset, which is a technique used with success in the past (9, 14). The idea is to distinguish between the left and right actuator effects by offsetting their continuous dither commands. Because of software limitations within the SRF VISTA environment, phase offsets in integer multiples of the sample time were all that could be accomplished. We did, however, demonstrate the improvement now shown in Figures 26 and 27. The rudder actuator, which has been historically troublesome and sluggish, showed surprisingly good results. Figure 28 shows detection within one second and good sustainment.

The longitudinal channel sensors show more of the same improvements. The only real trouble spot is the velocity sensor failure. Figure 29 shows that we've lost the ability to sustain a detected forward velocity sensor failure. This is not surprising when one considers the following items. First, the largest amount of pseudonoise added to arrive at this level of performance was placed directly into the forward velocity channel. As discussed earlier, we naturally expect desensitization to failures in that channel. Secondly, within the aircraft's state trajectory, failure insertion happens to take place just prior to the perturbation forward velocity variable recrossing zero. Since we fail sensors to their nominal (zero perturbation), nothing would be considered wrong, then, for a short period of time. At any rate, we attempt to compensate for this unacceptable performance in the next section and discuss the phenomenon further in Section 4.2.3. Figures 30 and 32 show good performance despite the desensitization within the longitudinal channel. Figure 31 reflects the result of our earlier pitch-channel/fully-functional tradeoff discussion.

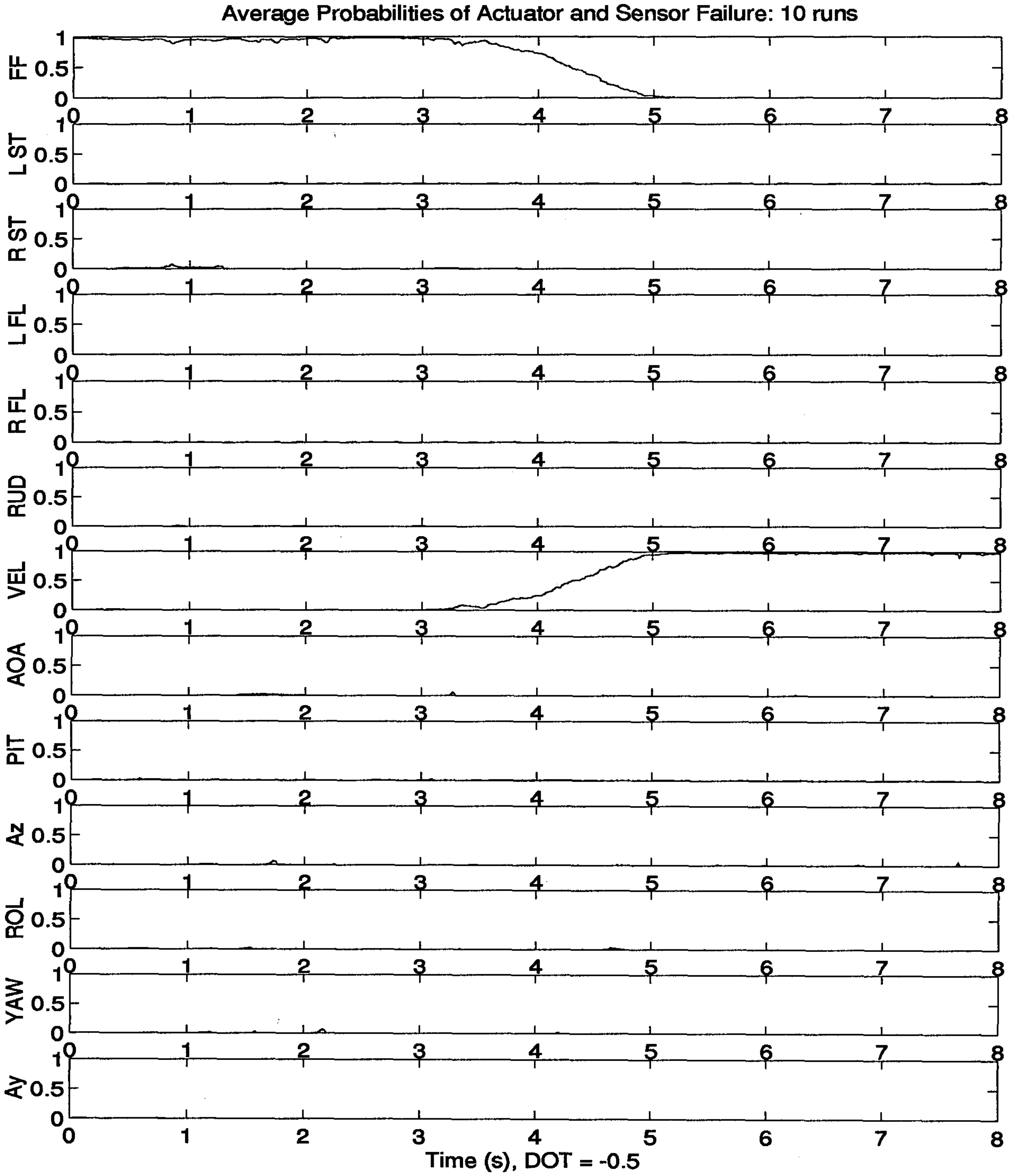
By comparison, performance of the lateral channel sensors is outstanding. Figures 33 to 35 show extremely fast response and good sustainment. This leads to the conclusion that focusing our attention on longitudinal equation of motion deficiencies was a good idea. It would seem that, once the dominating erroneous effects in the longitudinal channel have been removed, we observe very good performance within the lateral channel. One hypothesis we have for this is associated with the level of activity going on within each channel: much more in the longitudinal than the lateral. Another possibility might be the fidelity of simulation within each channel. More attention may have been spent including and coding longitudinal effects into the SRF VISTA F-16 simulation than lateral effects. This is understandable because most attention is traditionally given to longitudinal concerns such as speed and altitude, since they are often the dominant effects.

The preceding discussion and Figures 23 to 35 again show the performance improvement possible through direct addition of pseudonoise, with the tradeoff being increased desensitivity to actual failures. Whether or not this performance would be deemed acceptable depends, of course, on stated requirements. Logic schemes could be conceived which would set failure trigger levels to minimize false alarms. Slow performance such as resulted in the case of a failed pitch rate sensor may be quite acceptable in light of such a benign flight path. Even the fact that we miss the velocity sensor failure altogether may be acceptable if declaring it isn't critical at all. We wished to improve performance further however, so we sought to improve the velocity sensor failure with another technique.

Detuning R: Another way to admit uncertainty into our design model is in the update equations. Because measurements are imperfect, the seven sensors have each already been assigned a specific amount of uncertainty within the **R** matrix, the covariance of **v** in Equation (49). Since this effect is directly additive, we can increase or decrease uncertainty by tuning individual entries within **R**. In our implementation, there was some difficulty in detecting and sustaining a failure of the forward velocity sensor. We therefore looked to detune a bit on the element within **R** which corresponds to forward velocity. By doing so ($R_{vel} = 1 \times 10^{-4}$ versus the original value of $R_{vel} = 4.8 \times 10^{-6}$), we can improve performance, as shown in Figure 36. Within Figure 36 we see that performance improvement can indeed be gained using this technique. The probability is seen to take nearly two seconds to build up, but is sustained once there. Noting that forward velocity is not a flight critical measurement and that this "failure" is a zeroing of perturbations from the assumed nominal as opposed to a noisy indication of zero forward velocity, this performance can easily be characterized as acceptable.

Further performance improvement was gained during the algorithm's transient period of no-failure condition. Returning once again to Figure 36, one can easily note reduction in the magnitude of dropouts previously experienced in the first three seconds of the run (compare with Figure 29). Again, since this is the period of each run before the failure is inserted, it is improvement common to all of the single failure runs. Additional plots demonstrating this aren't included, to save space.

The cost associated with this technique is quite obviously speed of performance. A comparison of Figure 29 with Figure 36 demonstrates this nicely. Although the algorithm eventually loses the failure in Figure 29, the initial indication of failure is nearly instantaneous. With detuning present in **R**, the first indication of failure comes almost two seconds later. Clearly this is the tradeoff. Depending on application needs, one result may be preferred to the other. Our purpose is to demonstrate the possibilities this technique offers within this context of our application.



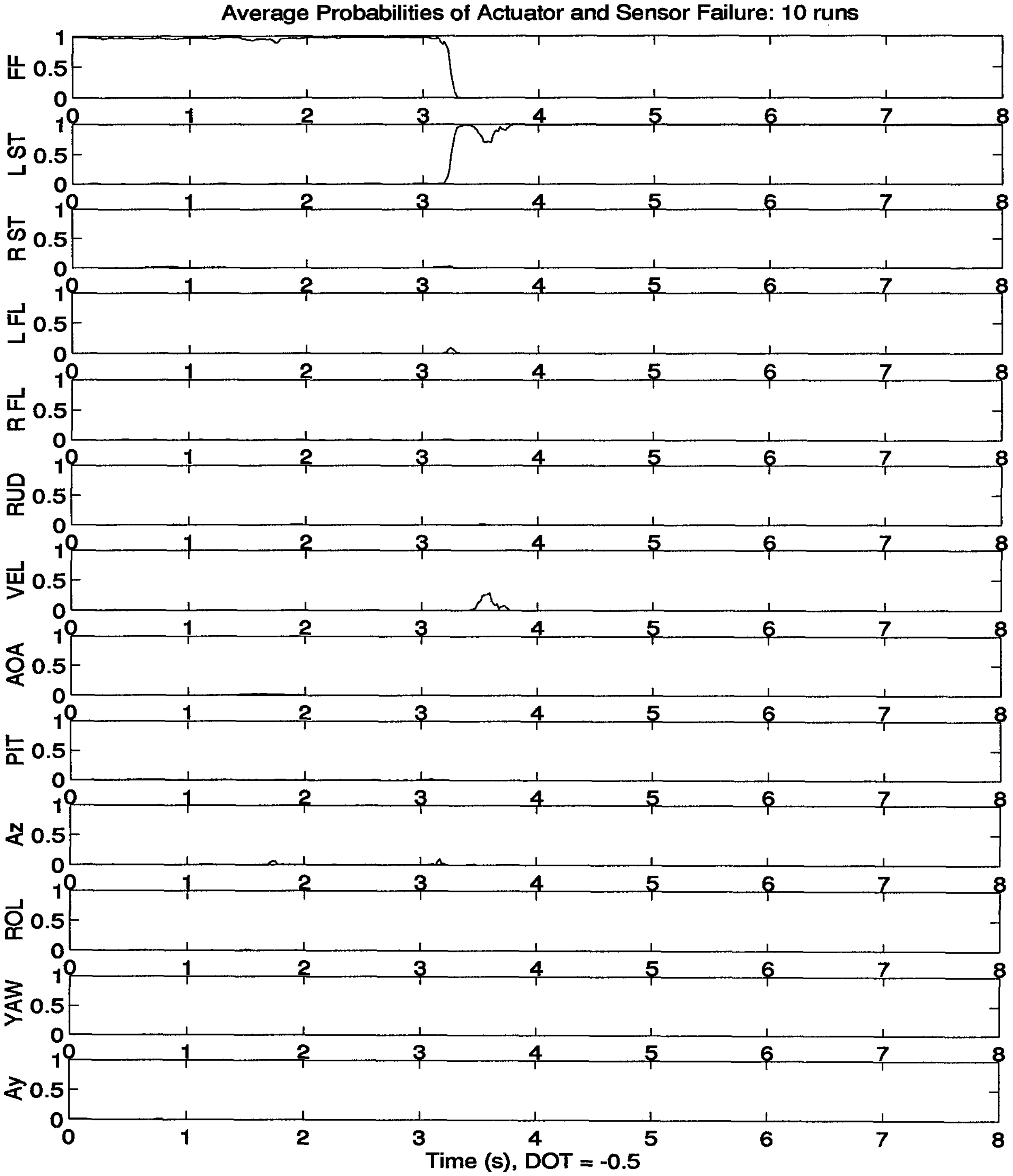
Notes: Filters = Q + long pseudonoise, Detuned R:
Menke Dither: Actuator Break = 14

Figure 36. Simulated *Forward Velocity* Sensor Failure with Longitudinal Pseudonoise and Detuning in **R**

Direct Pseudonoise on Lateral Q_d Entries: We now return to the bouncing behavior of the probabilities exhibited in Figure 24. In some applications this may be unacceptable due to the false alarm which arises in the yaw channel. This is unlike the forward velocity case just discussed, which dealt with missed failures. We therefore return to our covariance matrix for discretized dynamics pseudonoise, which currently contains only longitudinal entries. Now we focus on the lateral entries, specifically the yaw channel entry. Our aim is to detune directly on yaw much the same way as was done in the longitudinal channel with the hope of eliminating the bouncing shown in the left stabilator failure. Figure 37 shows the success of this technique using the following amount of pseudonoise:

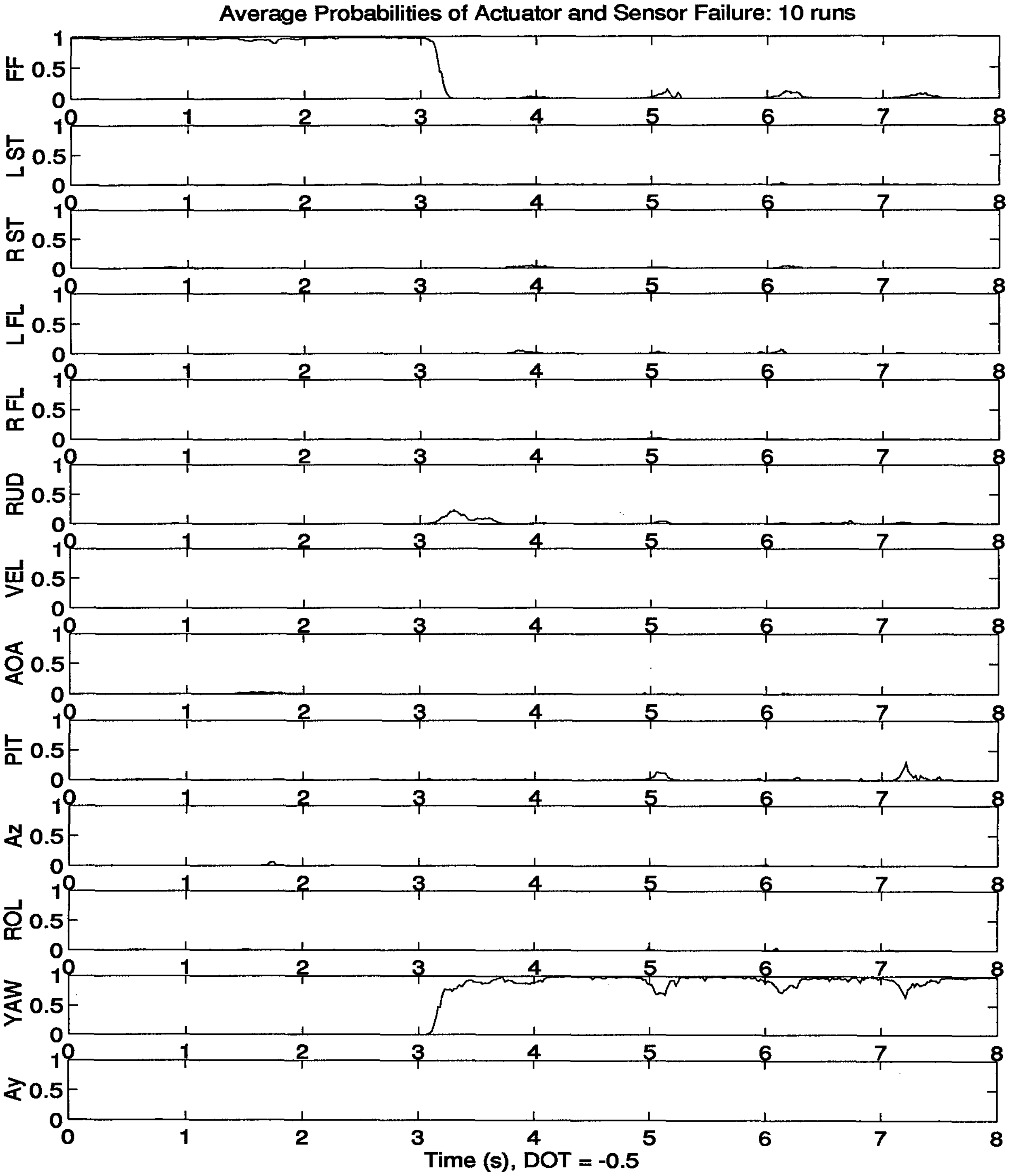
$$\mathbf{Q}_{d_{lat}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \times 10^{-6} \end{bmatrix} \tag{59}$$

All false alarms have been removed. Comparison of Figure 37 with Figure 24 shows that the firm indication of true failure has also been sped up from slightly over one second to about eight tenths of a second. Because we've detuned specifically in the yaw channel though, MMAE performance on yaw rate sensor failures and perhaps even rudder actuator failures may be expected to suffer. Figures 38 and 39 (compared to Figures 34 and 28, respectively) demonstrate the deleterious effects. What we gained in one area was lost somewhat in another. Again, the tradeoff is speed vs. stability. Only when firm specifications exist can we determine which performance is acceptable. This section does demonstrate however the kind of performance improvement which is possible again through direct addition of pseudonoise to the design model propagation equations.



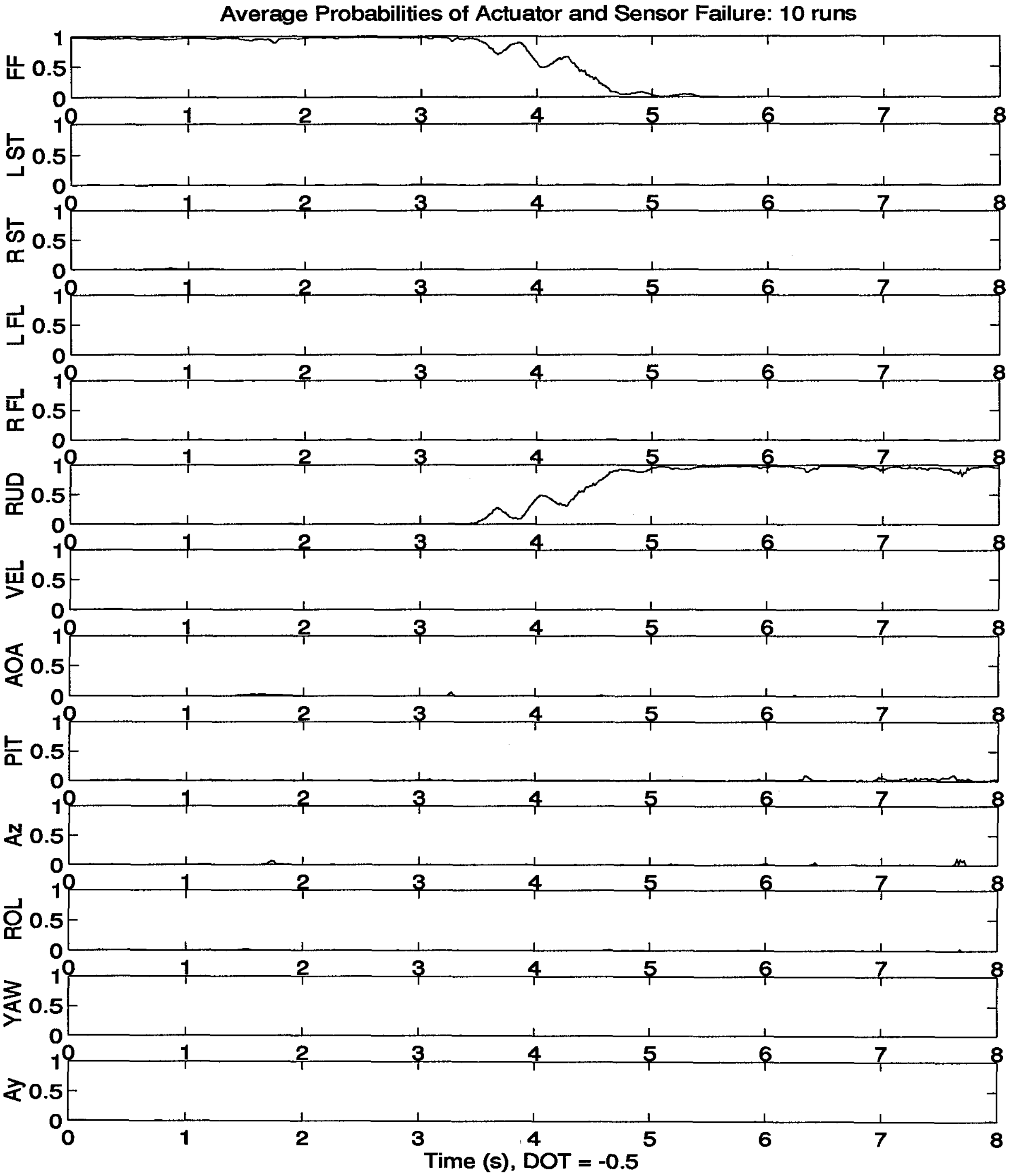
Notes: Filters = Q + long and lat pseudonoise, Detuned R:
Menke Dither: Actuator Break = 14

Figure 37. Simulated *Left Stabilator* Actuator Failure with Longitudinal and Lateral Pseudonoise and Detuning in $\mathbf{R}$



Notes: Filters = Q + long and lat pseudonoise, Detuned R:
Menke Dither: Actuator Break = 14

Figure 38. Simulated *Yaw Rate* Sensor Failure with Longitudinal and Lateral Pseudonoise and Detuning in **R**



Notes: Filters = Q + long and lat pseudonoise, Detuned R:
Menke Dither: Actuator Break = 14

Figure 39. Simulated *Rudder* Actuator Failure with Longitudinal and Lateral Pseudonoise and Detuning in **R**

Detuning for Actuator Uncertainty: A technique which we investigated but did not include in our final design attempted to compensate for a troublesome actuator. For unknown reasons, the results consistently show a probability drop into the right stabilator actuator channel during the first three seconds of unfailed operation in each run. This can be seen, for example, in Figure 23, but also appears in all of the other Figures (24 to 35) as well. Obviously, there is some predisposition within the MMAE towards declaring a right stabilator actuator failure at that point, probably caused by model mismatch. Our hope was to devise a way of reducing algorithm sensitivity to this failure mode only, in some manner similar to the tuning already demonstrated within $\mathbf{Q}_d$ and $\mathbf{R}$. The following equations show the strategy:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{G}\mathbf{w}(t) + \mathbf{B}_{actcol}w_B(t) \tag{60}$$

where $\mathbf{B}_{actcol}$ denotes the column of $\mathbf{B}$ corresponding to a particular actuator of concern, and $w_B(t)$ is the scalar pseudonoise to be added to that actuator. The corresponding covariance would then be:

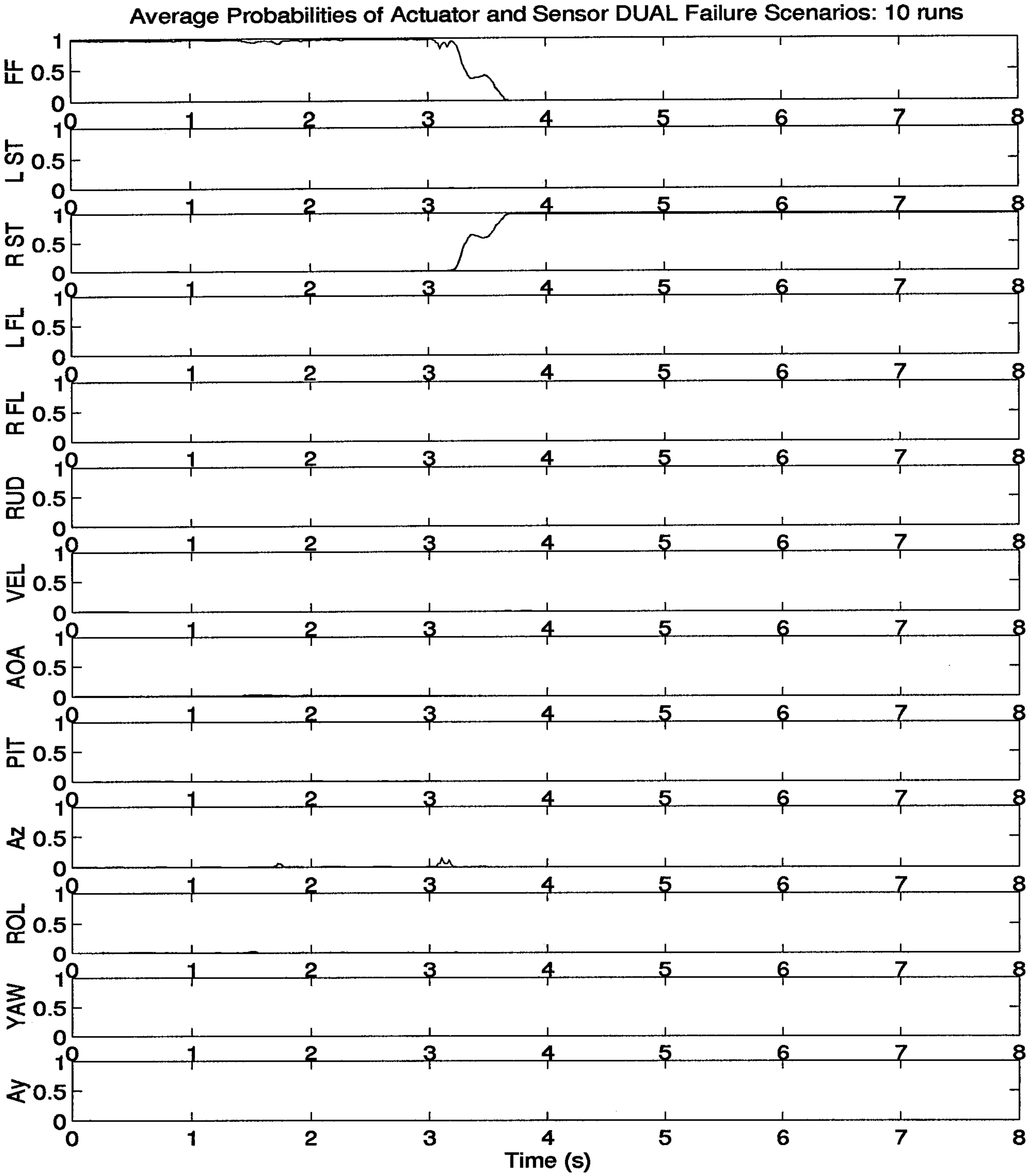
$$\dot{\mathbf{P}}(t) = \mathbf{A}\mathbf{P}(t) + \mathbf{P}(t)\mathbf{A}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T + \mathbf{B}_{actcol}q\mathbf{B}_{actcol}^T \tag{61}$$

where q is the strength of noise $w_B(t)$.

The idea was once again to bring a source of noise, source unknown, into the propagation equations at an appropriate entry point. This time however it should come in as an actuator effect. Equation (60) shows such a noise, in this case a scalar associated with a single troublesome actuator, being brought into the propagation equations through the correct column in **B** exactly as deterministic inputs enter. Equation (61) then shows the uncertainty entering as the tunable scalar noise strength being pre- and post-multiplied by the column of the **B** matrix associated with the troublesome actuator. We knew that deterministic inputs affect states through **B**; then we admitted uncertainty which we desired to enter the same way. Of course this technique could be expanded to handle multiple problematic actuators by creating a diagonal matrix of uncertainty and as many columns of the **B** matrix as appropriate, but in our investigation, we restricted attention to Equations (60) and (61). It is very important here to note that these equations reflect the unaugmented system dynamics of Chapter 3, Section 3.3.1. In practice, the column $\mathbf{B}_{actcol}$ must be stripped out of the augmented and **A** matrix since the noise $w_B(t)$ is intended to enter the system at an actuator output, not its input.

The results were encouraging. Figure 40 shows that, with such direct detuning on the right stabilator actuator, we removed the troublesome probability dip associated with it, while maintaining our ability to identify it when a failure truly occurs. A comparison of Figures 40 and 25 show the slight performance degradation cost involved in using this technique. As would be expected, we again lose speed of response in the failure for which we've detuned.

This technique is not, however, a part of our final design because its use further degraded sensitivity to the pitch rate sensor failure. Recall that our detuning in the longitudinal channel left us just able to identify this failure. Further addition of pseudonoise in any form, with no attempts made at optimizing our techniques, degraded the algorithm's *overall* performance. We therefore accepted the performance without this technique included and proceed with that design. The technique is quite viable, however, and is included for completeness and with the hope that it may find better application elsewhere.



Notes: Failure = R ST

Figure 40. Simulated *Right Stabilator* Actuator Failure with Additional Detuning for Actuator Uncertainty

Dither Adjustment: One final area worth mentioning for possible performance improvement is adjusting the dither strengths. As discussed in Chapter 2, the purpose of dither is to excite the system states during benign steady level flight. Without such excitation, failures become nearly impossible to detect. It was discovered however, that a tenuous balance exists between underexcitement and overexcitement: a problem which is only accentuated in this implementation by the Block 40 FCS. Overexcitation can lead to an aircraft violating its trim condition assumption which leads to false alarms, while underexcitation leads to missed alarms. A lot of interconnection exists between channels because of FCS features such as the Aileron to Rudder Interconnect (ARI). Attempting to get ideal excitation in all channels with a single dither scheme is nearly impossible. There is almost always a bit of overexcitation or underexcitation. For example, the dropout exhibited at the end of the run in Figure 23 is a direct result of overexcitation in the longitudinal channel. Bringing the dither down in this channel will remove the dip. Unfortunately, reduction of dither will result in underexcitation elsewhere. For example, bringing the dither down as just suggested reduced flaperon excitation and jeopardized sensing a failure in those actuators. Again, depending on the application and specifications, certain failures will be deemed more critical than others to detect. With such specifications in hand, an appropriate dither which correctly excites states can be generated.

Our research did not attempt to find an optimal dither scheme. Instead, as mentioned, we used the same dither as Menke for comparison purposes (26). Our conclusion is that dither must be tuned as a performance parameter from application to application.

4.2.1.3 Answers to Research Questions.

- *Do the elemental probabilities still converge to a solution?*

Yes. With the possible exception of the right flaperon, all thirteen failure hypothesis conditions converge to a solution. Depending on the triggering logic ultimately chosen to indicate a failure, sufficient differentiation exists between elemental filter probabilities in the failed right flaperon condition to declare convergence.

- *Is the solution the correct one?*

  Yes. All thirteen failure hypothesis conditions converge to the correct solution.

- *Are convergence times improved or degraded as a result of the higher order truth model?*

  Degraded. As anticipated, the MMAE experienced across-the-board increased difficulty declaring single failures as a result of the higher order and fully nonlinear truth model. Unlike Menke's results which consistently declared failures in less than one second, owing a lot to detuning, some failures take between one and two seconds before converging, due in great part to the detuning required.

- *Are there any heretofore unseen effects introduced as a result of the higher order truth model?*

  Yes. One problem already discussed and addressed was the higher rate of false alarms due to deficiencies within the design model. Without detuning, the reduced order design model does not take into account higher order unmodeled effects. The result is higher false alarm rates within the channels where the design model is deficient. Significant detuning was required to reduce the false alarms resulting from this effect.

  A second effect which revealed itself was an emergence of cross-axis ambiguities. Now that the higher order truth model includes cross-axis coupling, we experienced ambiguity across the longitudinal and lateral channels. A good example of this is the left stabilator failure already discussed. In Figure 24, we note that the ambiguity is not in the longitudinal channel, where a stabilator has its largest effect, but rather in the yaw channel. While this result is quite logical and to some extent expected, it is an effect which has previously been unseen. It has arisen from the higher order truth model and must now be taken into account during implementation such as demonstrated in this research.

Lastly, we point out the important role which trim variables play in performance. Although not apparent from the results presented, we found out "the hard way" during implementation that accuracy of the trim variables from which we calculate perturbations is critical. Any error in our declared trim is immediately picked up as an unexplained perturbation (δ = actual - trim) which can then be falsely declared as a failure. This intolerance to trim variable inaccuracy is a theme which is reiterated in Chapter 5.

4.2.2 Dual Failures. Having established a baseline filter design in the single failure scenario, the next logical step was to investigate dual-failure performance. Specifically, we wished to observe and characterize our algorithm's ability to detect and isolate two failures inserted into our new, nonlinear, higher order truth model. Previous research indicated that some sort of scheduling of appropriate dithers will be required to get good performance for all dual-failure combinations.

Our simulation induced the first and second failures at 3 and 5 seconds into the run, respectively, with the exception of when the rudder actuator or velocity sensor were the first failures. Recalling Figures 39 and 36, it took nearly 2 seconds for first failure detection. To ensure bank switch before the second failure was induced, we moved the second failure back to 5.7 seconds in these cases. The hierarchical approach described in Section 2.2.2.3 was used with the bank switching mechanism selected to trigger after one sample period of probability at 0.9 or higher. This choice (rather than require exceeding 0.9 for more than one sample period before switching) was based on the fact that all first failure ambiguity was removed in our single-failure detuning. Where ambiguities still exist, one would probably want to consider switching banks after multiple consecutive sample periods of failure indication.

All dual-failure combinations were exhaustively run and results qualitatively assessed. Knowing the level of performance obtained in the single-failure scenario, we appropriately appraised the MMAE's ability to detect and isolate the second failures. The results are provided in the next section.

4.2.2.1 Characterization of Dual Failure Results. Table 8 provides the results of this investigation. The possible ratings are "Good", "Fair", "Poor", and "ND". These ratings are given based on the following subjective criteria. "Good" results demonstrate second failure performance commensurate with, or better, than that achieved for a lone single failure. They are often characterized by a probability lock and hold. "Fair" results demonstrate performance somewhat degraded from when a lone single failure occurs. These are often characterized by significant dropouts from probability lock. "Poor" results demonstrate performance drastically degraded from when a single lone failure occurs. They are often characterized by some spiking in the correct channel, but no lock whatsoever. "ND" results indicate no detection. The second failure is completely missed or falsely declared. Appendix E contains twelve dual failure summary plots which represent a compromise between providing this table only and trying to include 132 separate plots.

The results indicate that in general, our dual failure performance is better than what has been demonstrated in past research (26). We attribute this to the detuning which was necessary in obtaining adequate single failure performance. Looking at Table 8, we note in general that, as might be anticipated, we get superior performance when the first failure is a sensor rather than an actuator. This is, in part, a result of the level of system excitation available for second failure detection, and is discussed further in the ensuing paragraphs. Also, since sensor failures are generally easier to detect quickly and solidly than actuator failures, there is less ambiguity at the time of second failure when the first failure is indeed a sensor. We can now focus our attention on the three primary trouble regions and discuss the possible reasons for these difficulties.

First Failure = Stabilator: Focusing on the first two rows of Table 8, we note that the algorithm struggles to identify the second failure when the first is a stabilator actuator. This can be attributed to reduced dither effectiveness. As discussed a few times previously, when the first failure is an actuator, the resulting loss of control authority diminishes dither effectiveness. This

	LS	RS	LF	RF	RUD	VEL	AOA	PIT	Az	ROL	YAW	Ay
LS		Good	**ND**	**Poor**	Good	Good	Good	**Poor**	Good	**Fair**	**Fair**	**Fair**
RS	Good		**Poor**	**ND**	Good	Good	Good	Poor	Good	**Fair**	**Fair**	**Fair**
LF	Good	Good		Good	Good	Good	Good	Good	Good	Good	Good	Good
RF	Good	Good	Good		**Fair**	Good	**Fair**	Good	Good	Good	Good	Good
RUD	Good	Good	Good	Good		Good	Good	Good	Good	Good	**ND**	Good
VEL	Good	Good	Good	Good	Good		Good	Good	**Fair**	Good	Good	Good
AOA	Good	Good	Good	Good	Good	Good		Good	**Fair**	Good	Good	Good
PIT	Good	Good	**Fair**	**Fair**	**Fair**	Good	Good		Good	Good	Good	Good
Az	Good	Good	**Poor**	**Fair**	Good	Good	Good	Good		Good	Good	Good
ROL	Good	Good	Good	Good	Good	Good	Good	Good	Good		Good	Good
YAW	Good	Good	**Fair**	Good	**Poor**	Good	Good	Good	Good	Good		Good
Ay	Good	Good	**Fair**	**Fair**	Good	Good	Good	Good	Good	Good	Good	

Table 8. Dual Failure Performance Summary Matrix: First Failure (row) vs Second Failure (column)

Rating	Qualitative Description
Good	Second failure shows performance commensurate with or better than for lone single failure. Look for probability lock and hold.
Fair	Second failure shows performance somewhat degraded from that of single failure case. Look for significant dropouts from probability lock.
Poor	Second failure shows performance drastically degraded from that of single failure case. Look for spiking of probability, but no lock.
ND	No Detect. Second failure completely missed or falsely declared.

Table 9. Dual Failure Performance Summary Matrix Key

can translate into insufficient state excitation for failure identification. Since stabilators are the primary control surface for the F-16, we would expect to see this effect most pronounced here (which it is). A possible solution is to schedule the dithering to account for the loss of control authority whenever the first failure is a stabilator actuator.

Another problem which occurred when a stabilator failed first was a consistent false alarm arising in the unfailed stabilator. This anomaly doesn't show up in Table 8 but can be seen in Figures 79 and 80 of Appendix E. The problem is thus described here. Approximately three quarters of a second after the correct bank switch which occurs after identifying the failed stabilator, a second, false, failure is declared in the opposite stabilator. At induction of the second failure, this false indication then gives way to the correct second failure. The fact that it gives way tells us it is a weak effect with a probability rise that is artificially inflated by the algorithm in the absence of a true second failure. A possible reason for this is yet another side effect of our detuning. The same indication of failure in one stabilator is enough to trigger the other when the definition between channels has been blurred by the detuning. Some logic may have to be employed to ignore the false alarms if this becomes a problem in other applications. Alternatively slightly-out-of-phase dithers might be applied to the two stabilators, in an attempt to disambiguate the two.

We also note different dual failure performance between the left and right stabilator, especially when the second failure is a flaperon. This is caused, again, by the phasing of the dither scheme. Recall the discussion in Section 4.2.1.2. The phasing of the three surfaces being dithered is such that, when a stabilator failure occurs, its secondary effects, like adverse yaw, either add into or out of the expected responses in other channels. Detecting the second failure when it is a flaperon is difficult enough because of the reduced state excitation just discussed. When the phasing effects work against detection, performance can get even worse. This phasing also explains why the **ND** shows up in opposite channels for opposite first failures, as shown in Table 8.

Second Failure = Flaperon: Columns three and four of Table 8 reflect general difficulty in picking up a flaperon as the second failure. There are a couple of good reasons for these results. First, however, a little background on flaperon difficulties will help set up the argument.

Although not reflected in this write-up, some time was spent in this research addressing the validity of our chosen dither scheme (The author hypothesized it to be a bit violent for this implementation). In tinkering with the control surface deflection magnitudes, it was noted that the flaperons and rudder cannot be separately worked at our desired levels because of the aileron-to-rudder interconnect (ARI) within the F-16 FCS. The ARI exists to provide the pilot with "feet on the floor" coordinated turning capability. The upshot is that, at our current level of excitation, the flaps are providing minor amounts of control authority whose effects are further minimized by the ARI. With that in mind, we address the specifics.

When the first failure is a stabilator actuator, the difficulty of identifying the second failure when it is a flaperon just becomes that much more exacerbated. This combination of deleterious effects accounts for the majority of our "ND" ratings. In fact, another effect which doesn't appear in Table 8 or Appendix E, is that this combination of failures often false alarms into the rudder channel. Again, ARI involvement is probably to blame.

The other tough spots seem to occur even when the first failure is a sensor. We feel that this is because the particular sensors - pitch rate, normal acceleration, roll rate, yaw rate, and lateral acceleration - contain important indications of flaperon failure. Recall that the effects of a failed actuator must propagate through the system and show themselves as abberant behavior in some or all of the sensors. If indications of flaperon failure are already known to be slight at the onset, loss of important information contained in any of the mentioned sensors may be enough to degrade performance significantly. If flaperon failures are deemed critical, more work will be required in this area to overcome the ARI problem.

Rudder Actuator/Yaw Rate Sensor: This combination of failures has historically been troublesome because there is only one rudder and the effects of the two failures are so uniquely entwined. Take the first case in which the sensor fails first, then the actuator. With no sensor available to sense the missing yaw rate, performance is bound to be poor. The loss of rudder must somehow be picked up through the little information coming through other sensors which measure other variables. Now consider the other case, in which the actuator fails first and then the sensor. Without any yaw rate being generated (adverse yaw is negligible), the algorithm cannot possibly know that the sensor has just failed. Thus, we note the only other "ND" rating observed. A possible solution to this problem would be to add a position sensor to the rudder. This third piece of solid information would alleviate the ambiguities we've noted.

A final point which wasn't a trouble spot *per se*, but deserves further discussion, is the change we made in the time of second failure insertion for the cases of first failure being rudder actuator or velocity sensor. As stated, this was done because, on average, it took about two seconds to identify either of these first failures. On average, of course, indicates that some individual failure runs took longer than two seconds to be identified. In these cases, the second failure was being induced before the first failure was identified and bank switch had taken place. Often, the second failure was then identified before the first and the bank switch was not the one we had in mind. At the end of the run, however, it became a moot point because both failures were always identified correctly regardless of which bank came on line first. The problem was that it adversely corrupted our dual failure plots, from which erroneous conclusions about performance could be drawn. We therefore chose to push the second failure induction back in these cases from 5.0 seconds to 5.75 seconds to ensure that all runs had already switched to the bank corresponding to the first failure.

Not shown explicitly here is the ability of the hierarchical model to back out of bank levels when failures disappear. This was investigated though and confirmed in the course of our research. Exclusion of a detailed demonstration of this ability in the write-up is intentional as an attempt to save space. Suffice it to say, when "backing out" was required, it was achieved consistently to yield the good performance reflected in Table 8. While no attempt was made to improve upon these results, we've identified the problem areas and, where possible, offered viable solutions. Pursuit of such improvements ought to be accomplished within the context of any future applications.

4.2.2.2 Answers to Research Questions.

- *Do the elemental probabilities still converge to solutions?*
- *Are they correct?*

For over 95% (126 out of 132) of the dual failure combinations, yes. The algorithm understandably has some difficulties; when the first failure is a stabilator because of the loss of dither effectiveness, when the second failure is a flaperon for the same reason and that its control authority effects are being fought by the aircraft to rudder interconnect (ARI), and when the failure combination is rudder actuator/yaw rate sensor because they have such similar effects. For all other cases investigated, probabilities converged to the correct solutions.

- *Are convergence times path dependent?*

Although we didn't specifically address this in the write-up, a glance at Table 8 and the plots in Appendix E will show that indeed, there is some path dependence. This is easily understood as the result of position within the flight trajectory at failure induction, as well as how much state excitation remains when the second failure is induced.

- *Are convergence times improved or degraded as a result of the higher order truth model?*

In general, they were degraded. The effects noted as a result of detuning in the single-failure scenario are equally applicable here. There were numerous cases, however, where convergence times were within a few sample periods; at least as fast as anything demonstrated in the past (26).

- *Are there any heretofore unseen effects introduced as a result of the new truth model?*

Yes. A very nice surprise was that, in many cases, the dual failure initial performance with the higher order truth model was superior to the same cases reported for a lower order model (26). We attribute this result to the filter detuning done to achieve adequate performance in the single-failure scenario. The improvement seems to have also spilled over to second failure identification. Further tuning for specific applications may remove the need for a dither scheduling scheme altogether.

4.2.3 Time of Failure Insertion Variability Demonstration. Returning to the forward velocity sensor failure of Section 4.2.1.2, we wished to pursue the issue of the *time* of failure insertion further. In Section 4.2.1.2, we determined that the reason this particular failure mode was slow to converge was because failure insertion took place as the perturbation variable crossed zero. Intrigued by this finding, we chose to prove the point by rerunning the forward velocity sensor single-failure scenario with insertion taking place at different times along the nominal trajectory.

Figure 41 summarizes the results of this demonstration. The figure consists of eight plots. The first plot displays the nominal trajectory of the forward velocity perturbation state variable. It is included as a reference and to assist in making our point. The seven succeeding plots each display the forward velocity failure hypothesis filter, over ten Monte Carlo runs, with the forward velocity sensor failure inserted at 1, 2, 3, . . . , and 7 seconds.

The plots rather dramatically demonstrate the effect that timing of failure insertion can have on performance. At 3.0 seconds, when the perturbation is at or very near zero, there is inherent ambiguity (recall that a hard failure is defined to be a zeroed sensed perturbation variable). Actually, this ambiguity between true failure and perturbation zero in the trajectory exists for all sensor variables. It is only a problem however, with the aircraft's "slow variables" such as forward velocity and angle of attack, when they hover near zero perturbation for any length of time. The other sensed variables undergo a higher rate of change such that the don't stay for any appreciable time at zero perturbation. Looking again at Figure 41, we note that, once velocity has moved away from zero perturbation, identification is nearly instantaneous. This can be attributed to the

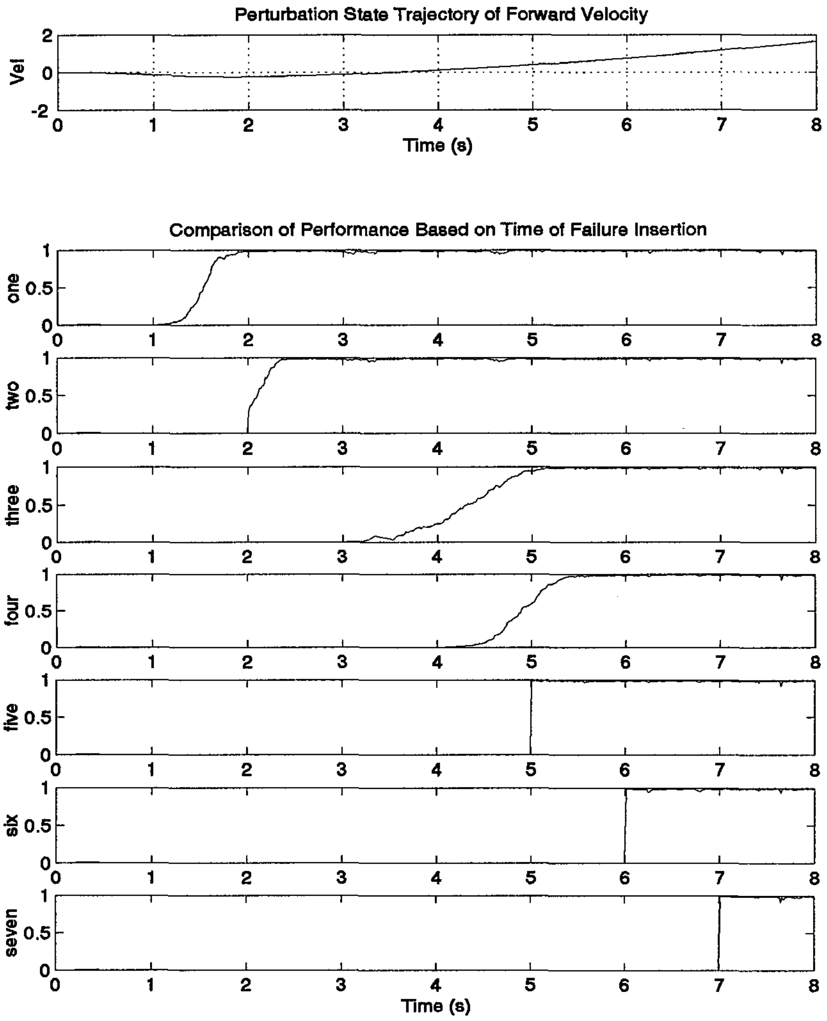


Figure 41. Time of Insertion Variability Demonstration

fact that, as the aircraft moves further away from the nominal design point, it is more inclined to declare failures in general. Being thus closer to the failure declaration threshold, when a true failure does occur, it is identified more quickly.

The difficulties we've uncovered here are inherent to the algorithm. An MMAE or similar algorithm will always have difficulty identifying a slow variable sensor failure when the failure occurs near zero perturbation. In Chapter 5, we take this logic one step further.

4.3 Residual Monitoring

Recalling Equation (17), we had expected the residuals of the filter with a correct failure hypothesis to be zero-mean, white, Gaussian, and of covariance $\mathbf{A}$ ($= \mathbf{H}\mathbf{P}^{-}\mathbf{H}^{T} + \mathbf{R}$). Menke demonstrated that an additional failure declaration vote could be cast based on the behavior of scalar residuals using simple tests such as "σ-bound", "zero-meanness", and "whiteness" tests. (26). We wished to re-investigate the behavior of the individual filter residuals in light of the new host environment to determine if such additional corroborative votes still exist.

4.3.1 Characterization of Residual Monitoring Results. The single-failure software developed in Section 4.2.1 was used in this investigation. For every single failure induced, time histories of the seven residuals from each of the thirteen filters were recorded. Figures 42 to 53 document the results. In each, the fully functional filter's residuals appear in the left-hand column of plots and the residuals corresponding to the filter correctly hypothesizing the induced failure (induced at 3.0 seconds in this simulation) are in the right-hand column. The results clearly show that residual monitoring, for this application, will not be as effective an additional voter as was the case in Menke's work.

The first aspect lost was the measure of quantitative certainty provided by the "σ-bound" test. Recall that the residual covariance matrix $\mathbf{A}$ is calculated as shown above, and that theoretically, the scalar residual values in a well matched filter should fall within its 2σ-bounds 95% of the time. In our implementation, however, we can no longer use this test. The heavy detuning required to achieve adequate probability convergence has rendered a 2σ-bound quite meaningless. Model mismatch was compensated by increasing the magnitude of entries within the $\mathbf{R}$ and $\mathbf{Q}_d$ matrices. This both directly and indirectly ($\mathbf{Q}_d$ through $\mathbf{P}^{-}$) drove up the entries within $\mathbf{A}$ and in the process, removed the theoretical basis on which the "σ-bound" test is based. The σ-bounds have been omitted from the residual plots as a result.

A second area of some degradation is in the actuator failure residual monitoring. Menke's models were so well matched that good residual behavior could be observed within some of the actuator hypothesis filter residuals. Recall that this is predicted for *well matched* filters and also that the actuator effects propagate indirectly through all sensor residuals. This is the most difficult residual monitoring scenario and, as we've seen throughout our development, our models are no longer so well matched. Looking at Figures 42 to 46, we can observe only occasional indications of failure like those previously discussed. Flaperon failures in Figures 44 and 45, for example, show clear indication of failure in the lateral acceleration (Ay) residual. The magnitude of the dither oscillation shrinks considerably at the onset of failure. A better example appears in Figure 46. In this case, the yaw residual quite nicely exhibits zero-mean "whiteness" and a cessation of oscillation when the failure is inserted. Unfortunately, no similarly strong indicators exist for the stabilators, as shown in Figures 42 and 43, although the Ay residual does show detectable changes in magnitude and mean value when the failure is induced. The problem though, is that although these indications of failure do exist, with the exception of the rudder failure, they are slight and would certainly require more elaborate tests than the ones proposed to get consistent additional votes.

Looking again at Figures 42 to 46, we can get some idea of why we have observed this degradation. Remember that most of our detuning took place in the longitudinal channel. Restricting our attention, then, to the longitudinal channel residuals (VEL, AOA, PIT, and Az), we see failed filter hypothesis residual behavior which in almost all cases, mirrors the fully functional hypothesis filter residuals. This is directly a result of detuning. The filters are so detuned, that their behavior and mismatch begins to resemble that of the filter based on the fully functional hypothesis. Only in the lateral channels observed, Ay and YAW, have any of the characteristics of closely matched filters been preserved. Further, the heavy detuning is also responsible for the fully functional hypothesis residuals not behaving as expected when no failures exist. The mismatch is big enough that, for the most part, predictable behavior is lost, although observable degradation often appears when the failure is inserted. This is observable in all of the figures (42 to 53). If our earlier hypothesis concerning the level of detail involved in the lateral channel of the SRF simulation is true, our difficulties here will be further exacerbated by more mismatch in the real world. The implication of all this is that we've lost our good additional voter sensitivity to actuator failures. This is not the case for sensor failures, however.

Figures 47 to 53 show that we've retained our sensitivity to sensor failures, although not quite in the same manner as in the past. Good residual characteristics show up within the scalar residual corresponding to that failure (sensor) and not in all scalar residuals of the filter with the correct failure hypothesis, as theory predicts. This, again, is due to the fact that detuning masks the effects in the other residuals, but the direct effects in the failed residual are strong enough to come through. Remember, this is the easiest of the residual monitoring scenarios and indeed, is the idea upon which residual monitoring is based. Looking then at the residual plot pairs corresponding to the failed sensor in Figures 47 to 53, we see that, while the "σ-bound" test has been lost, the "zero-meanness" and "whiteness" tests remain quite valid. In Figure 47, for example, upon failure insertion at 3.0 seconds, the failed hypothesis residual, VEL, becomes both zero-mean and white. It would be a good candidate for the "zero-meanness" test. In Figures 48 to 53, the disappearance of oscillations, the onset of zero-mean "whiteness", and reduction of magnitude to reasonable levels, could help to generate a vote. Two of the original simple tests, testing for dither frequencies, and perhaps a qualitative version of the "σ-bound" test, remain valid for generating an additional vote for sensor failures.

Of note is the oscillatory behavior discussed in Chapter 2, Section 2.2.4. In many of the mismatched filters, particularly the ones identified as candidates for monitoring, we observed oscillatory behavior during mismatch. Unlike past research, though, we a couple of distinct frequencies. In the fully functional VEL residual of Figure 47, we see the characteristic dither frequency. The failed hypothesis PIT residual of Figure 49 also shows this oscillation, but in addition, shows a lower frequency oscillation. This slower one can be attributed to an aircraft natural mode oscillation in the longitudinal channel (phugoid) triggered by our dithering. Similar modal oscillations are present in the lateral channel too, as seen in the failed hypothesis YAW residual of Figure 52. These additional frequencies were not noted in past research because the modes weren't included in the reduced order models. The SRF simulation contains these higher order modes which show themselves for exactly the same reasons as explained for the dither oscillation in Chapter 4, Section 2.2.4. Knowledge of their presence as well as their frequencies, which can easily be detected using the data gathered from the other tests, may help in future additional voter logic schemes.

These results clearly show that scalar residuals still contain useful failure information (more so for sensors than actuators). Without the quantitative "σ-bound" test, the challenge will be to utilize the remaining available tests in such a way as to provide an additional failure declaration vote reliably. To regain some of the lost confidence in the "σ-bound" test, we propose that the "whiteness" and "zero-meanness" tests be performed on both the candidate residuals and the corresponding fully functional filter residuals. A qualitative comparison of the residual pair's characteristics, which we expect still to be quite different, might provide this additional confidence. The simple magnitude comparison test, like the "σ-bound" test, might also be tried. This could conceivably provide a simple enough test to attempt residual monitoring on the failure indications noted for the actuator failures too.

Even though we've lost our theoretical advantage in approaching residuals, information still exists from which additional failure status votes might be cast. Depending upon the application, it appears that residual monitoring can still be an effective tool in declaring sensor failures. Beyond that, a tradeoff would have to be considered which pits good probability convergence against nice residual behavior. In this research, priority went to probability convergence with the resulting residual behavior just presented.

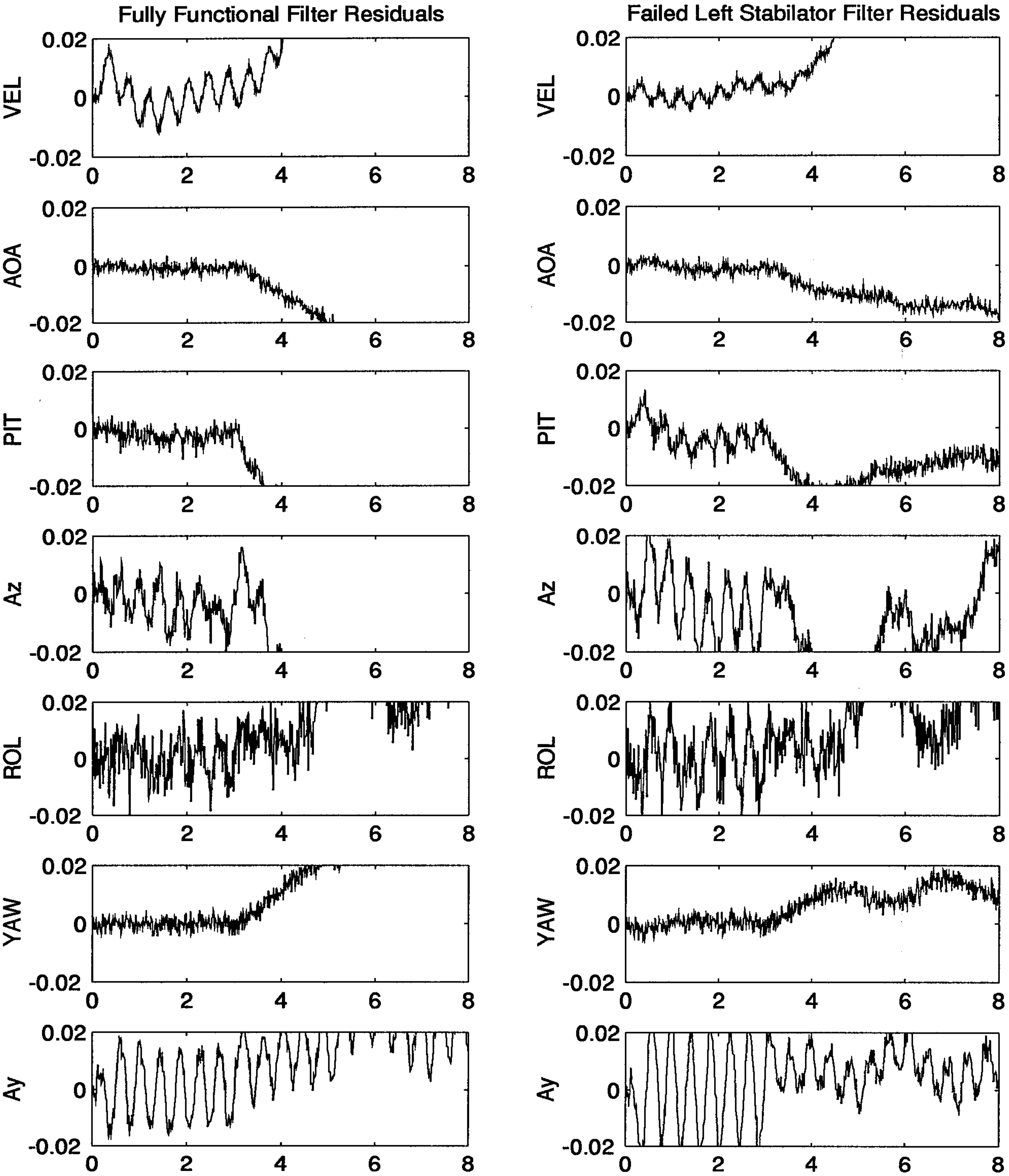


Figure 42. Comparison of Fully Functional and Left Stabilator Filter Residuals: (Left Stabilator Actuator Failure Induced at 3.0 seconds)

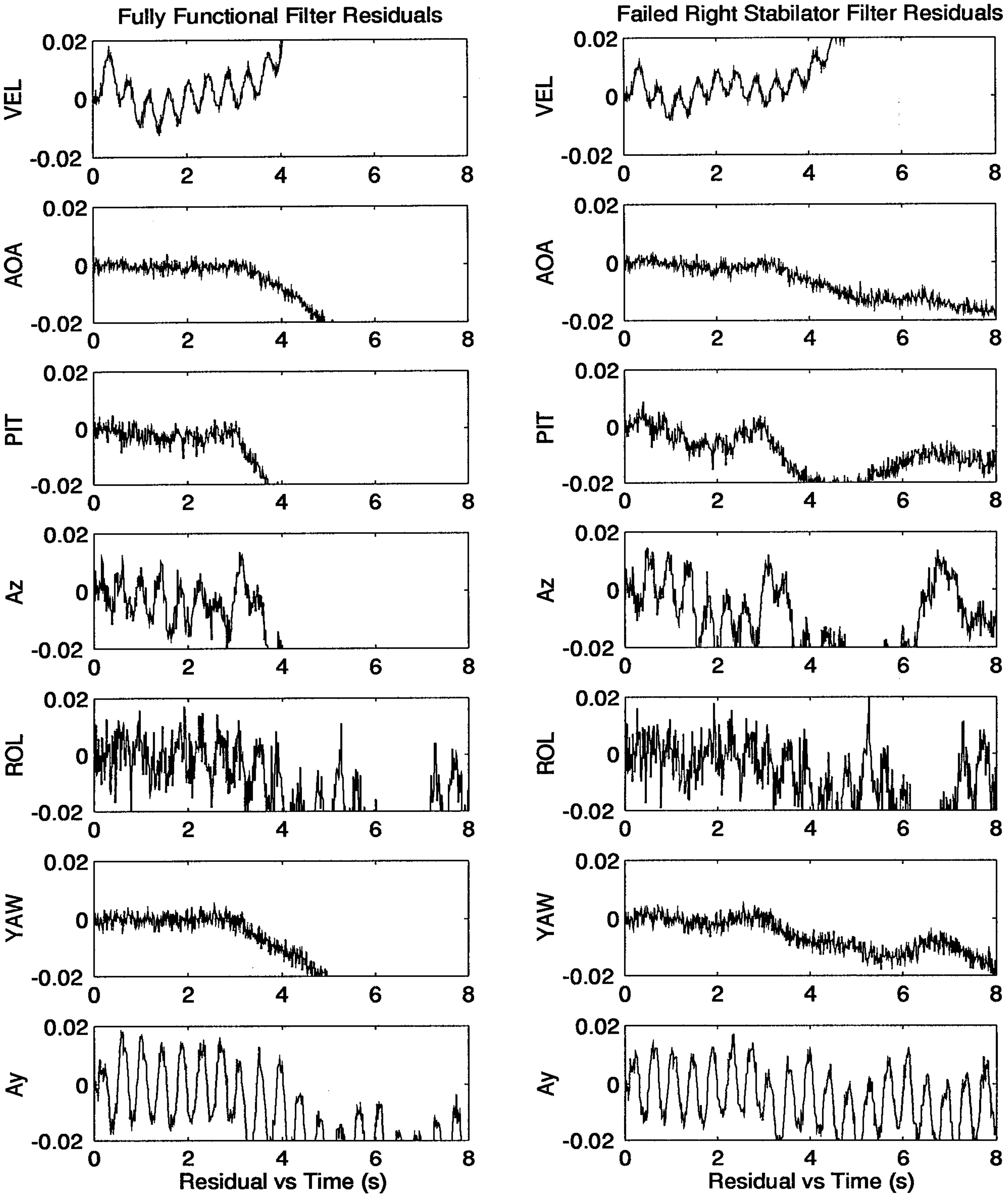


Figure 43. Comparison of Fully Functional and Right Stabilator Filter Residuals: (Right Stabilator Actuator Failure Induced at 3.0 seconds)

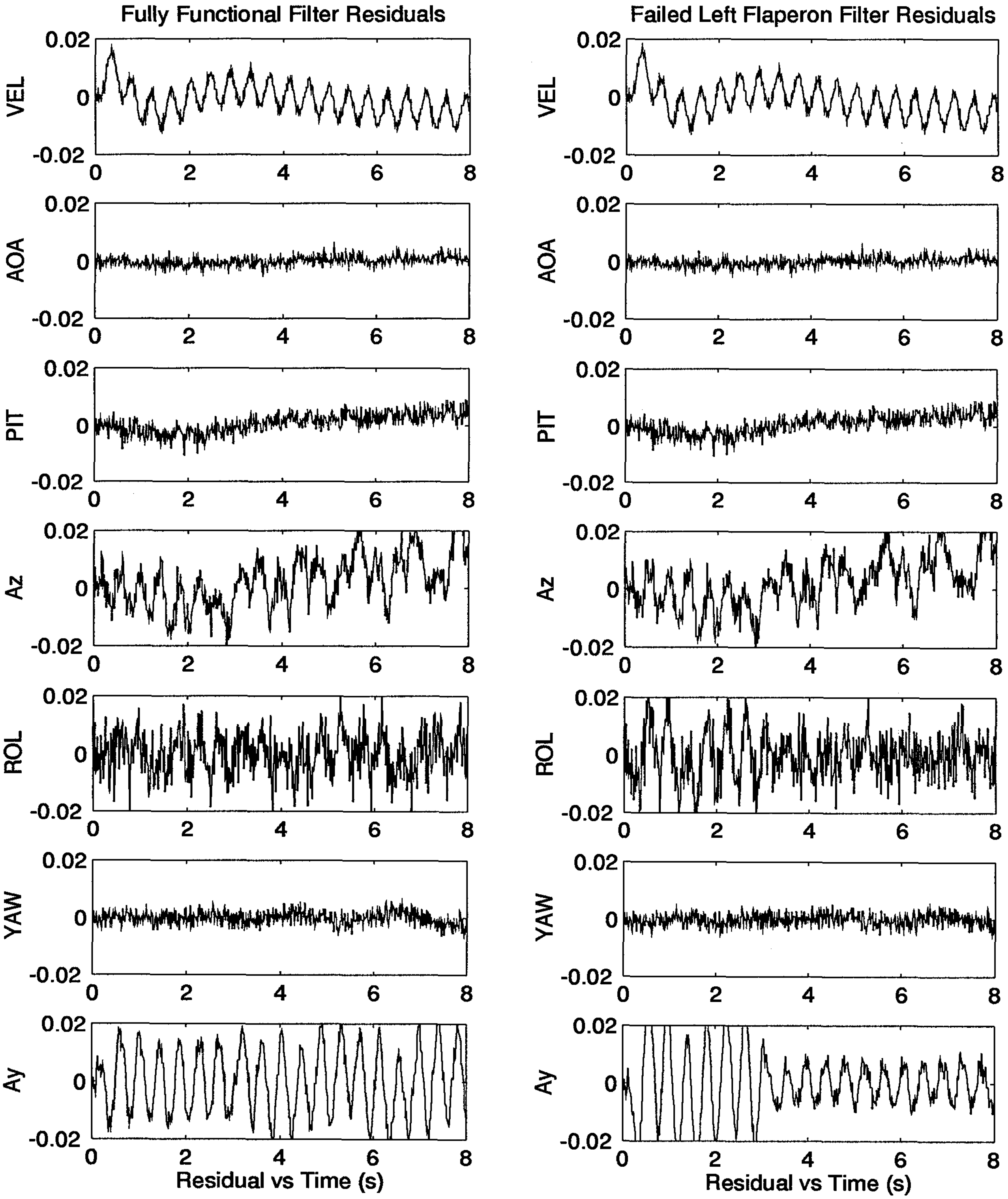


Figure 44. Comparison of Fully Functional and Left Flaperon Filter Residuals: (Left Flaperon Actuator Failure Induced at 3.0 seconds)

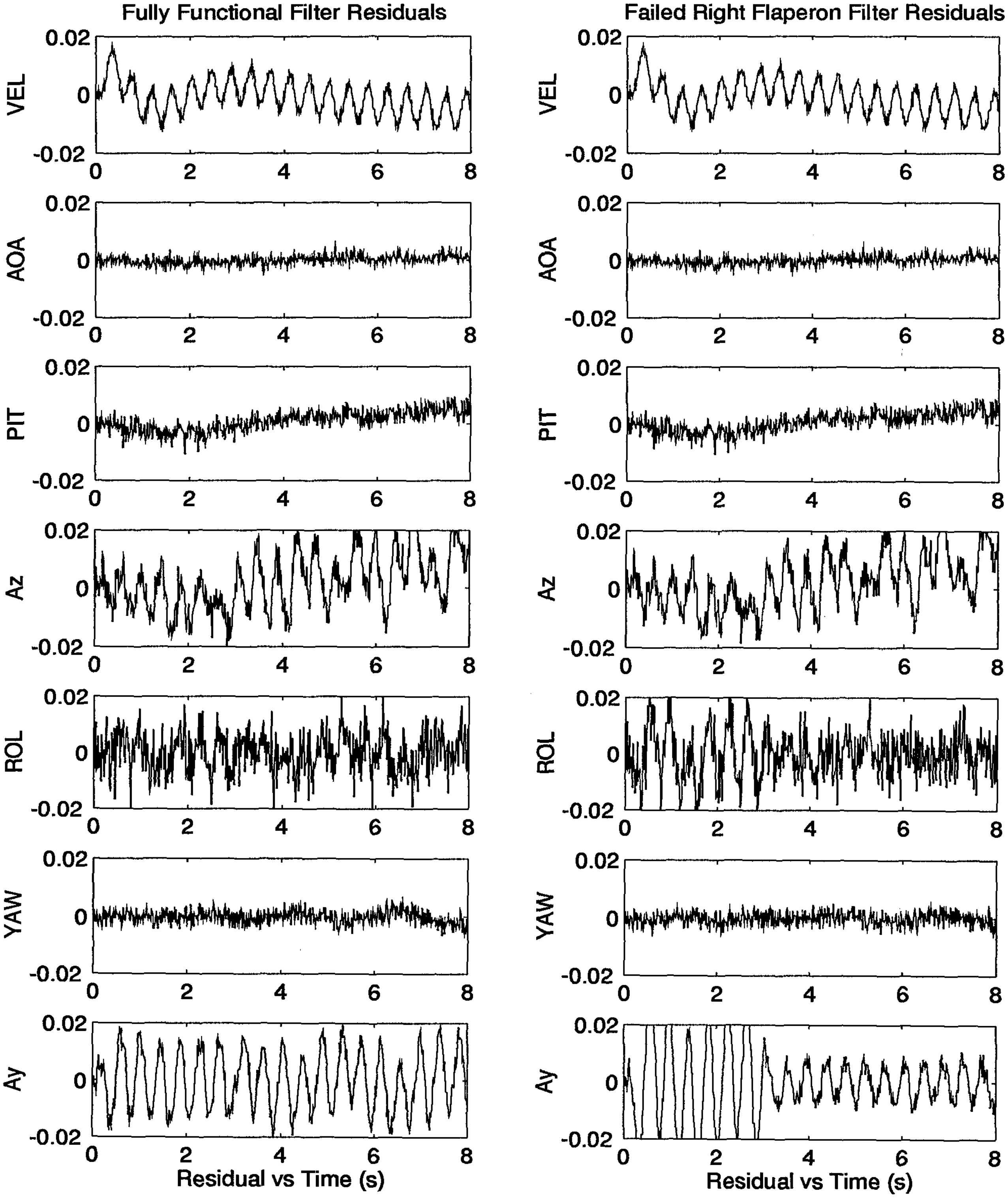


Figure 45. Comparison of Fully Functional and Right Flaperon Filter Residuals: (Right Flaperon Actuator Failure Induced at 3.0 seconds)

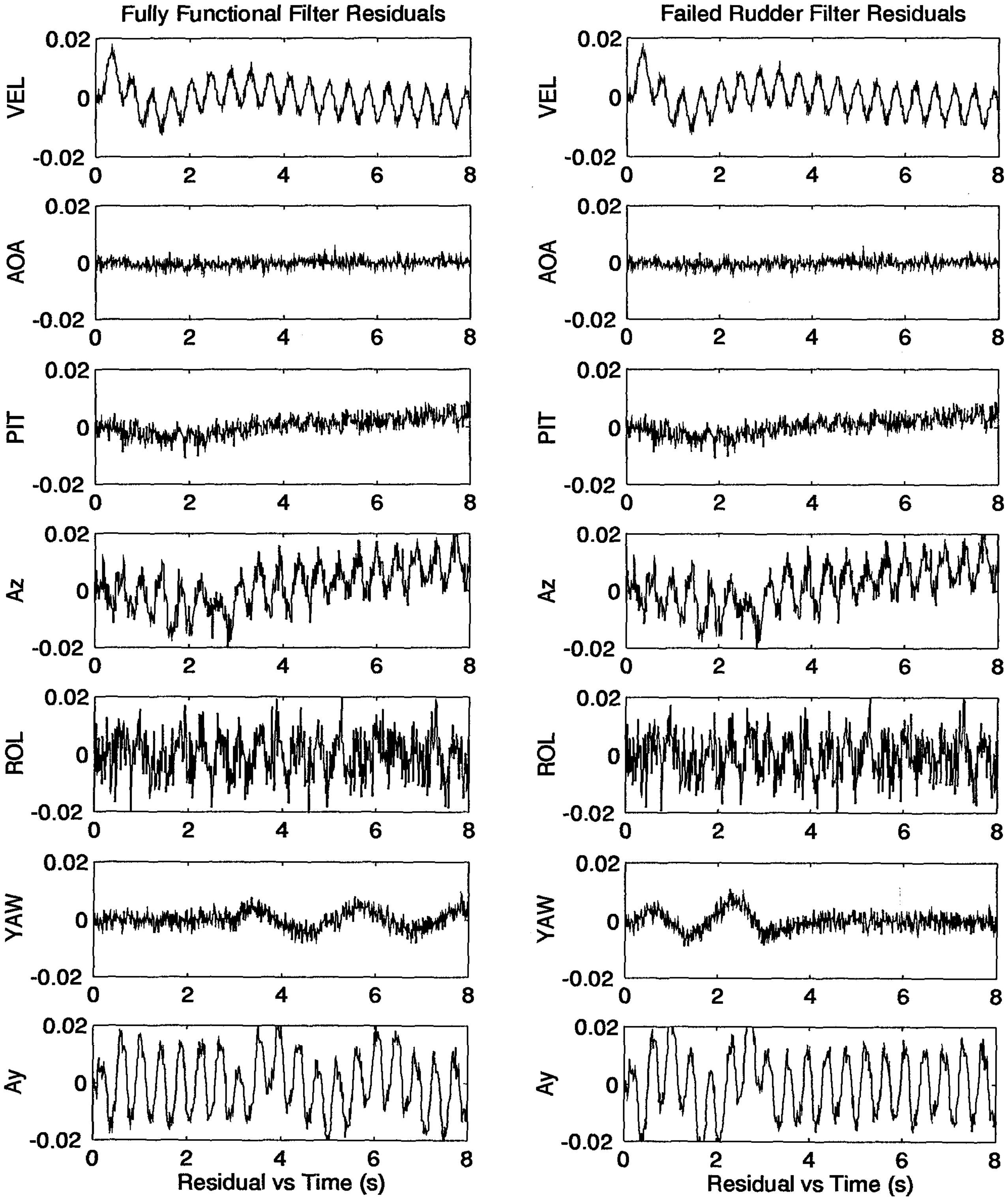
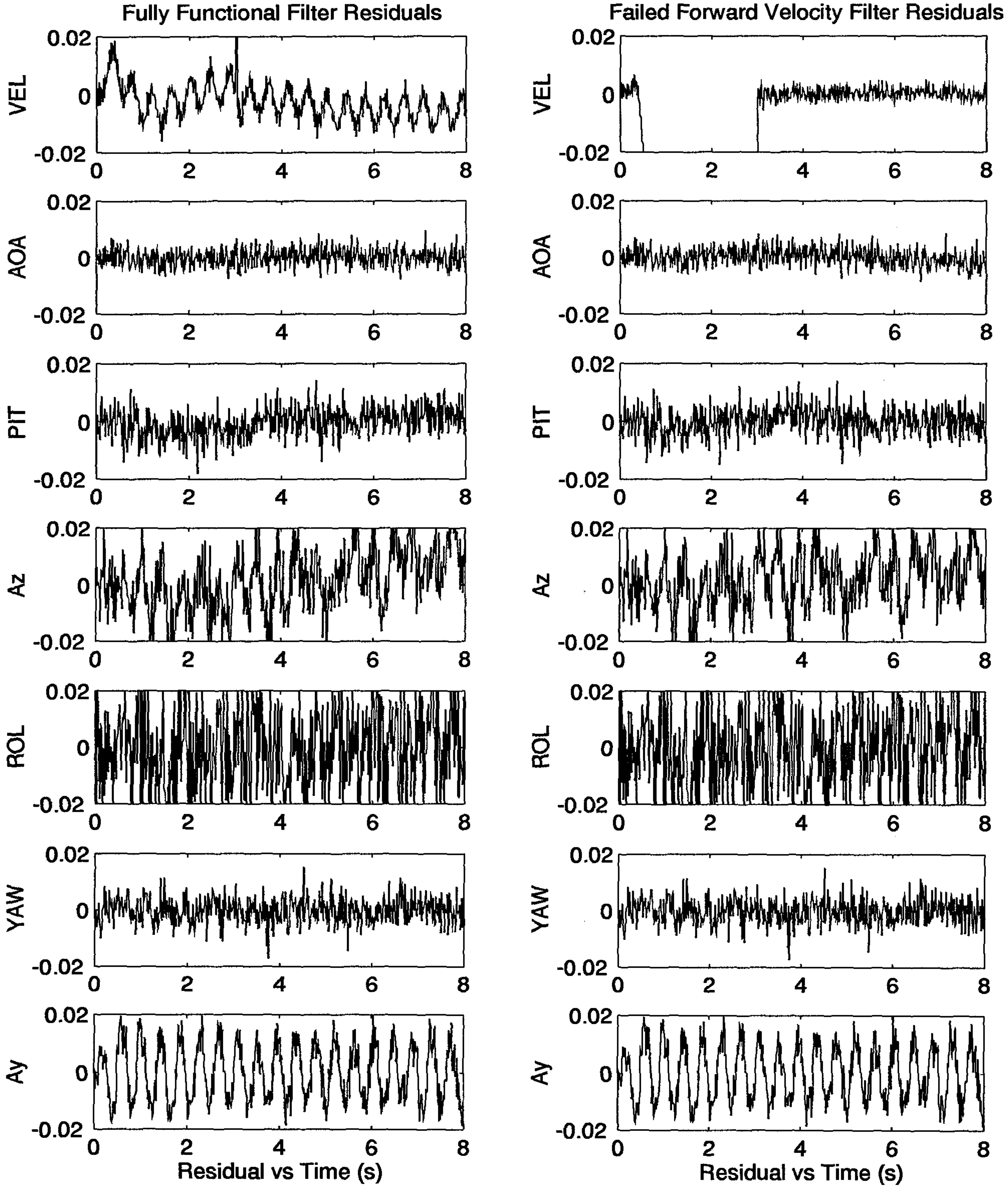


Figure 46. Comparison of Fully Functional and Rudder Residuals: (Rudder Actuator Failure Induced at 3.0 seconds)


Fully Functional Filter Residuals

Failed Forward Velocity Filter Residuals

Figure 47. Comparison of Fully Functional and Forward Velocity Filter Residuals: (Forward Velocity Sensor Failure Induced at 3.0 seconds)

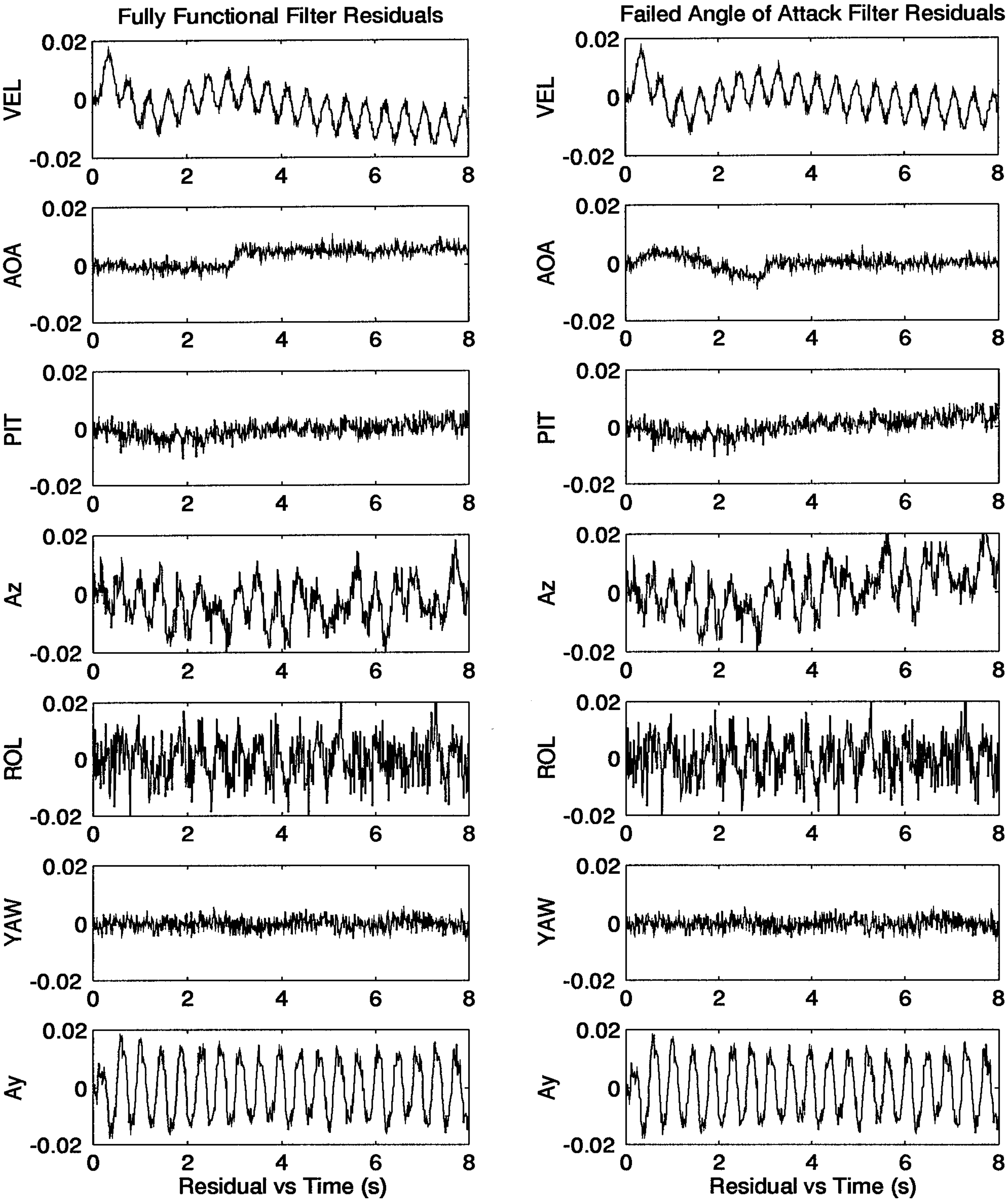


Figure 48. Comparison of Fully Functional and Angle of Attack Filter Residuals: (Angle of Attack Failure Induced at 3.0 seconds)

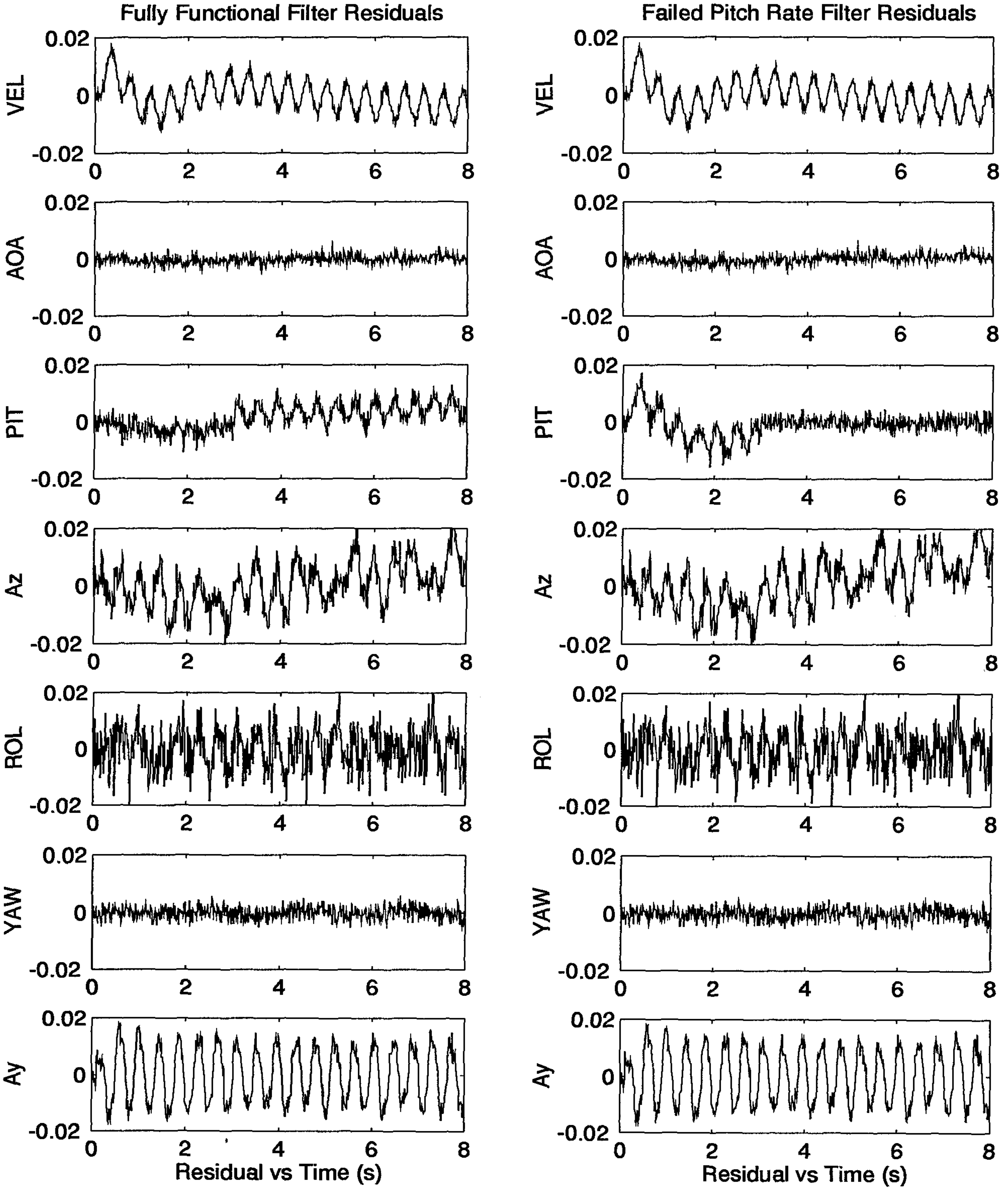


Figure 49. Comparison of Fully Functional and Pitch Rate Filter Residuals: (Pitch Rate Sensor Failure Induced at 3.0 seconds)

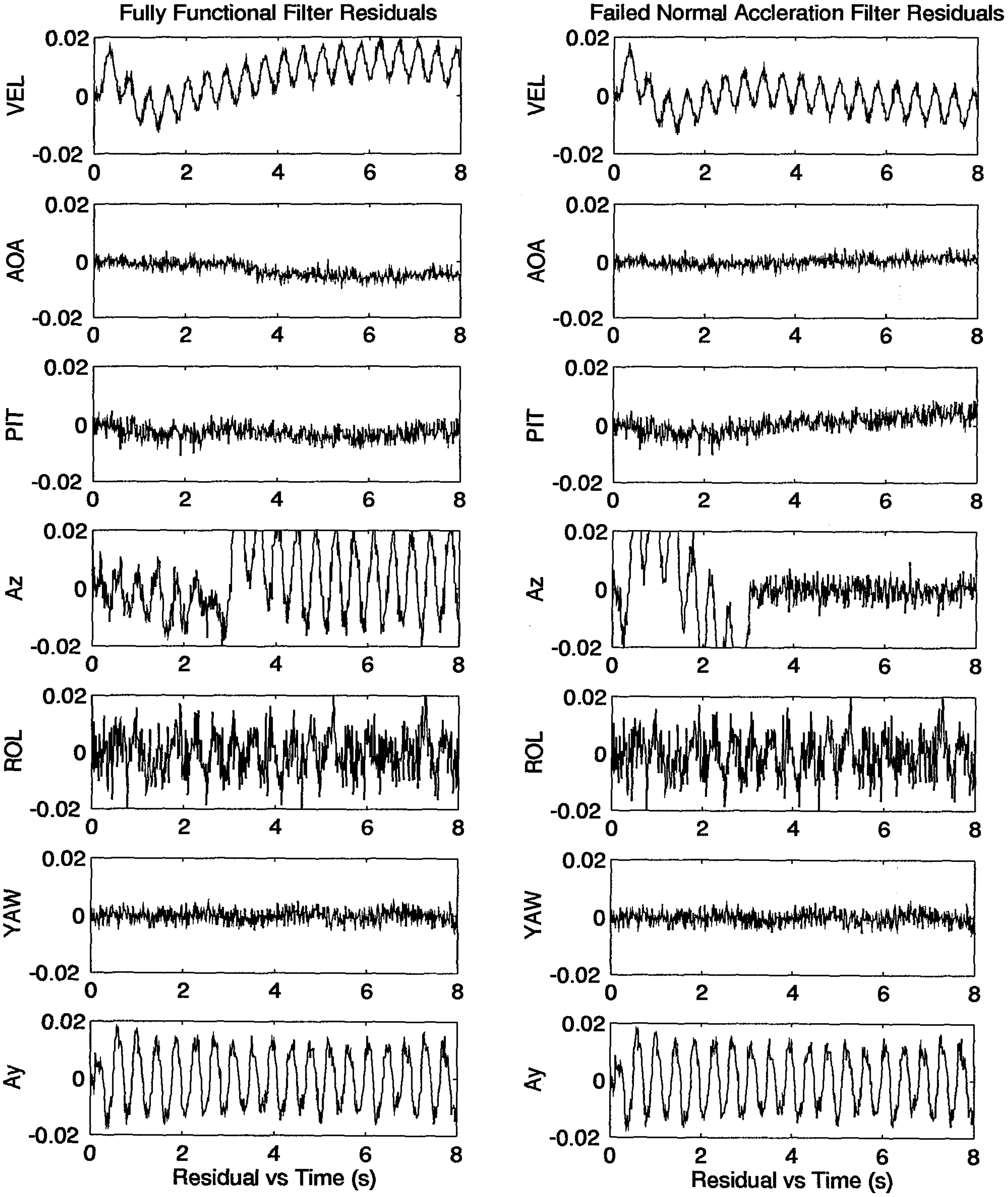


Figure 50. Comparison of Fully Functional and Normal Acceleration Filter Residuals: (Normal Acceleration Sensor Failure Induced at 3.0 seconds)

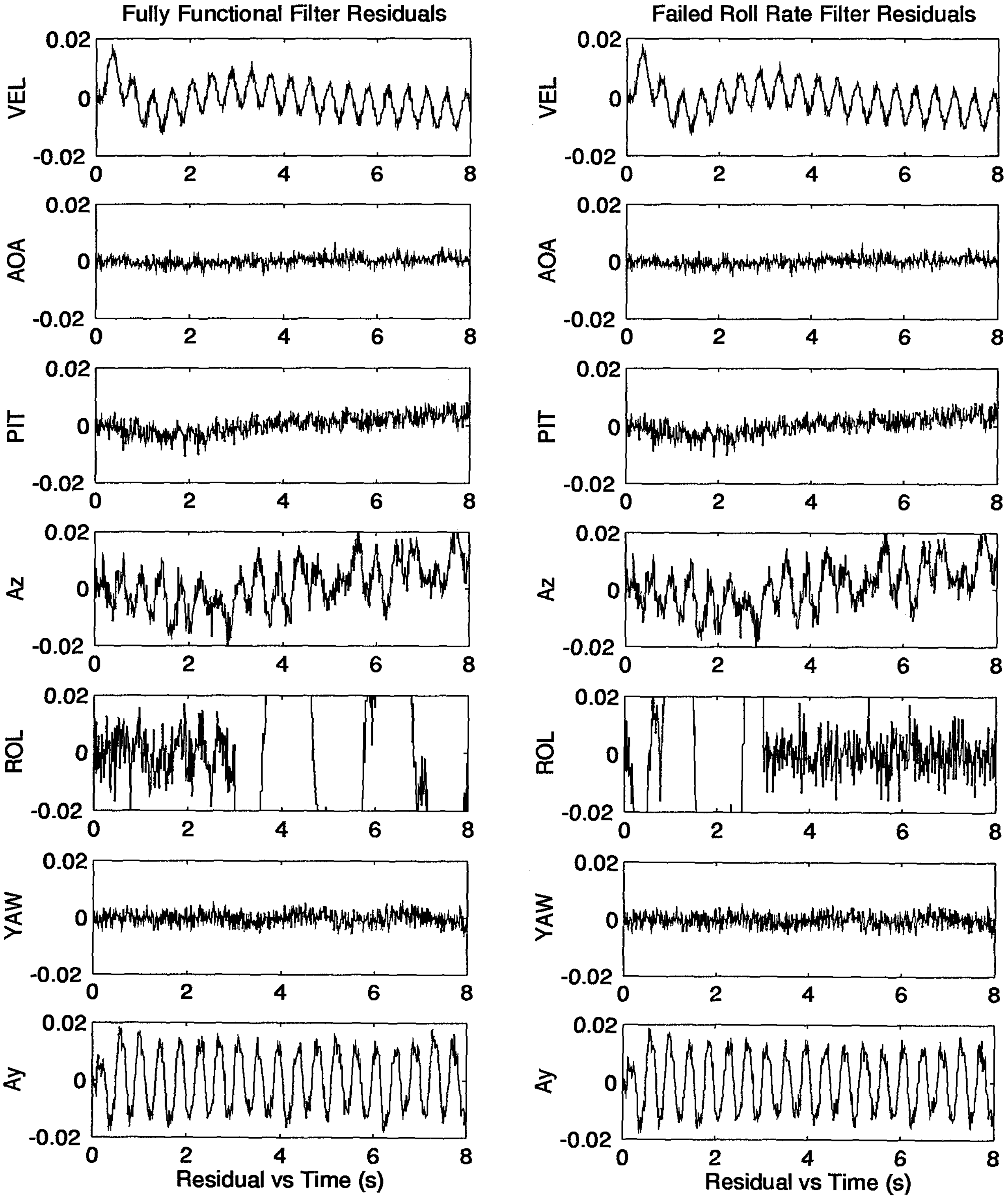


Figure 51. Comparison of Fully Functional and Roll Rate Filter Residuals: (Roll Rate Sensor Failure Induced at 3.0 seconds)

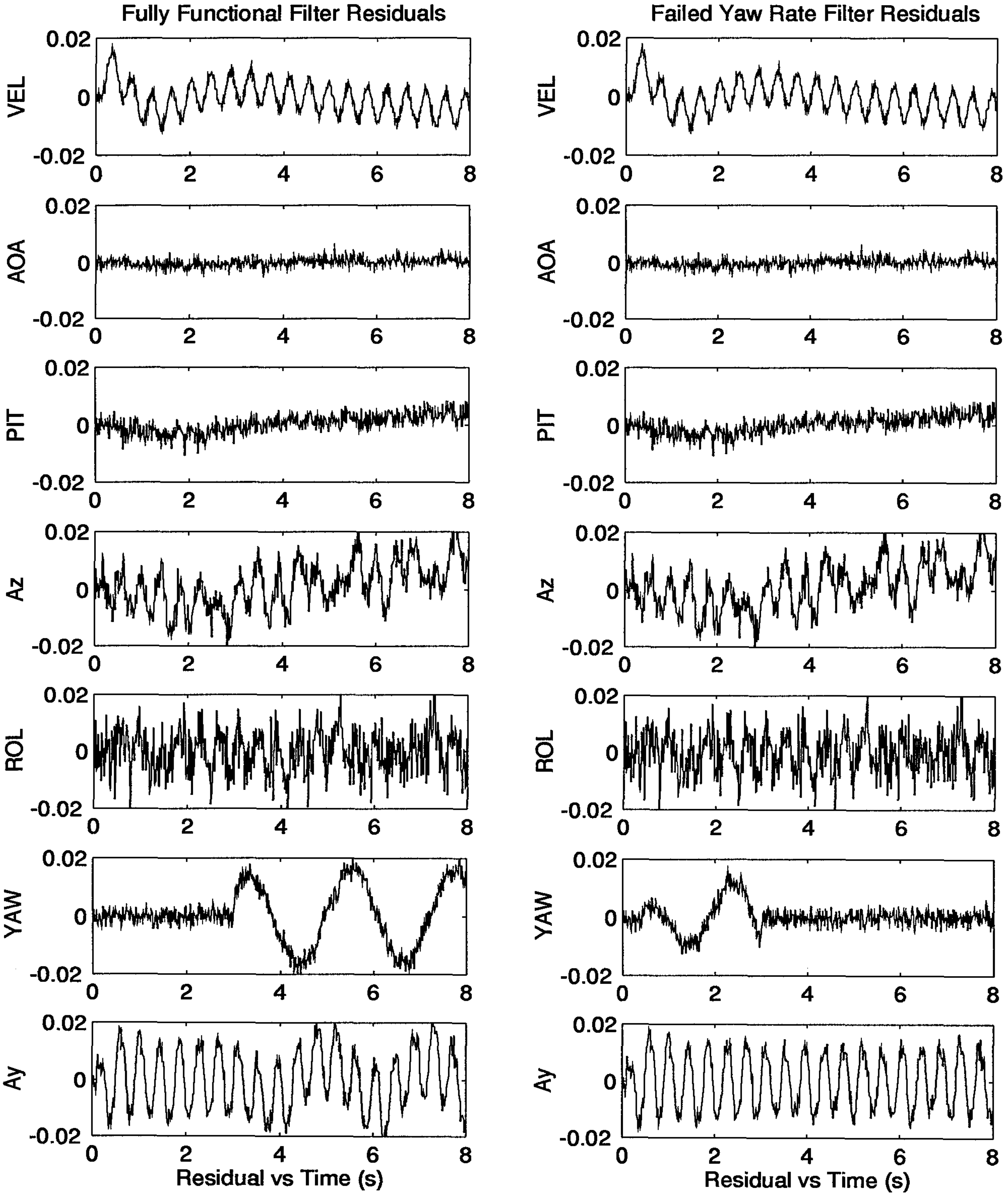


Figure 52. Comparison of Fully Functional and Yaw Rate Filter Residuals: (Yaw Rate Sensor Failure Induced at 3.0 seconds)

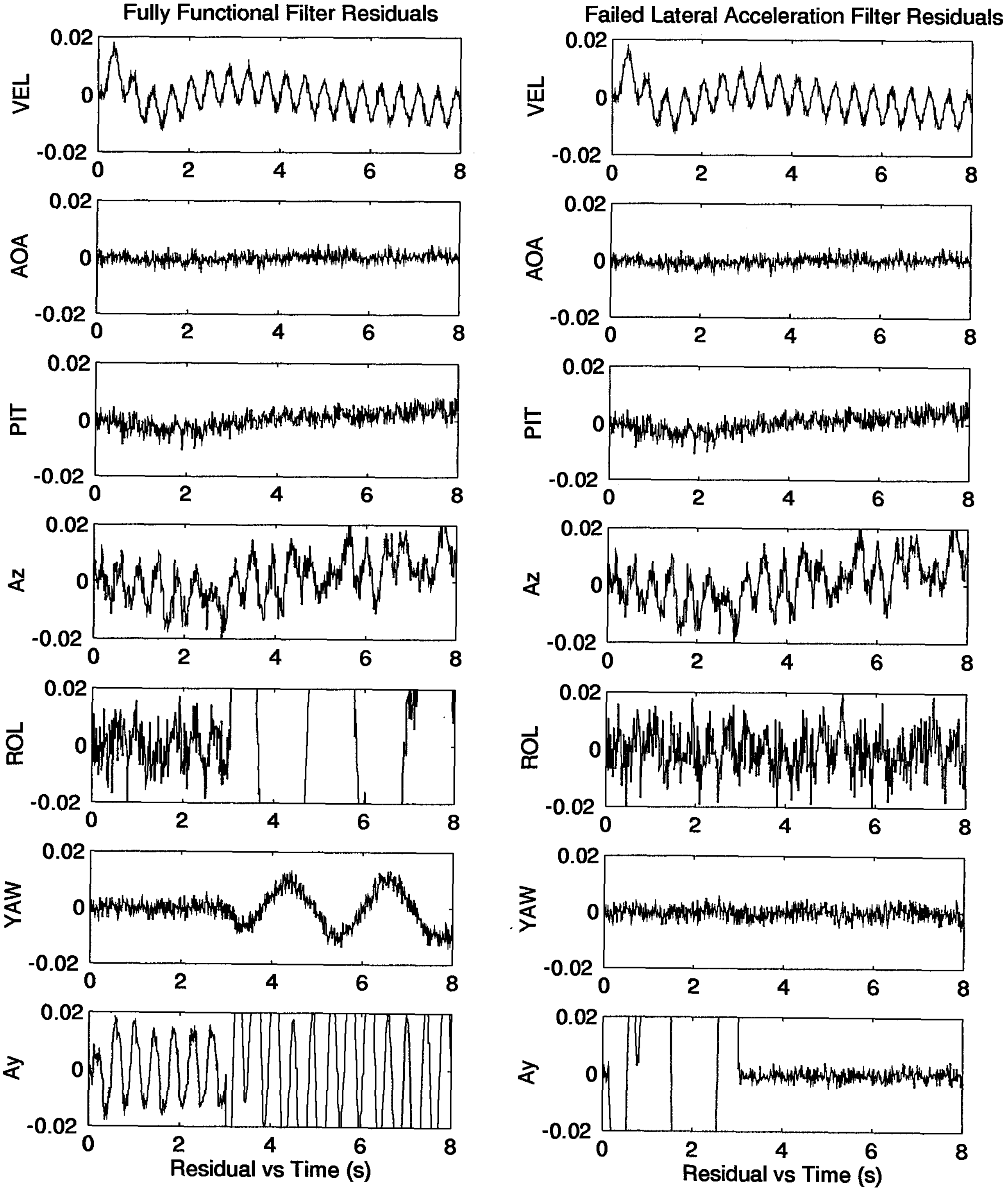


Figure 53. Comparison of Fully Functional and Lateral Acceleration Filter Residuals: (Lateral Acceleration Sensor Failure Induced at 3.0 seconds)

4.3.2 Answers to Research Questions.

- *Can residual monitoring be used as an additional voter for sensor failures?*

Yes. As just demonstrated, there is enough information from which one could determine the aircraft sensor failure status and possibly the actuator failure status as well.

- *Can residual monitoring be used as an additional voter for actuator failures?*

Possibly. Detuning has substantially masked the effects which actuator failures may have on sensor-generated residuals, but with more sophisticated qualitative tests, the failure information may be reliably extracted.

- *Are there any heretofore unseen effects introduced as a result of the new truth model?*

It is unclear how the higher order truth model directly affects performance. One way perhaps is the appearance of aircraft natural frequencies within mismatched filter residuals. A more indirect way might be the performance degradation we've attributed to detuning. One could argue that the need for detuning was a result of the design model's mismatch with the new truth model. Since filters were not tuned specifically for good residual characteristics though, it is difficult to say with any certainty what the direct contribution of the new truth model was to the residual characteristics' degradation.

4.4 MMAC Demonstration

While no dedicated efforts were expended developing a Multiple Model Adaptive Controller (MMAC), the benefits of one were very demonstratable with the MMAE-based control algorithm already in place. In the case of *sensor* failures, MMAE-based control already provides the robustness promised by the MMAC. The following paragraphs demonstrate this capability.

The figures provided depict the time histories of the aircraft's basic state vector with single failures being induced at 3.0 seconds. In all plots, the nominal trajectory, or in other words, the path of an unfailed aircraft, is superimposed as a dotted line. It is our contention that, if the trajectory of an aircraft with a failure induced follows the fully functional profile, then the FCS was insensitive, or robust, to that failure.

Figures 54 to 58 correspond to actuator failures. MMAE-based control is **not** insensitive to this type of failure, and it shows. The stabilators and rudder show significant departure at the onset of failure because they provide much of the aircraft's control authority. Flaperon deviation is also apparent, but at a level commensurate with their usage. These five plots nicely demonstrate the ramifications of not being robust to failure.

Conversely, Figures 59 to 64 demonstrate the potential of MMAC. In all cases, the MMAE's internal estimate of state is being used to reconstruct the critical sensor values, which are then fed into the FCS rather than inputting raw sensor data. Not only does this provide a noise-free measurement, but also accounts for the insensitivity to sensor failure. Note though, that the performance in the longitudinal channel is somewhat degraded as a result of our heavy detuning in that channel. Slight deviations are noted in the plots associated with longitudinal variable (Figures 59 to 61), while trajectories are nearly exact in plots associated with lateral variables (Figures 62 to 64). This again is because the accuracy of the filters within the detuned channels have been compromised somewhat as a result of the detuning. The plot associated with a forward velocity sensor failure have been omitted from this report because FCS simulation software precluded our using the MMAE-provided signal. In fact, the F-16 FCS doesn't use a direct measurement of forward velocity in its gain scheduling, but rather employs static pressure which is measured by other sensors (pitot tubes). Our MMAE estimate of forward velocity is therefore not input to the FCS in MMAE-based control.

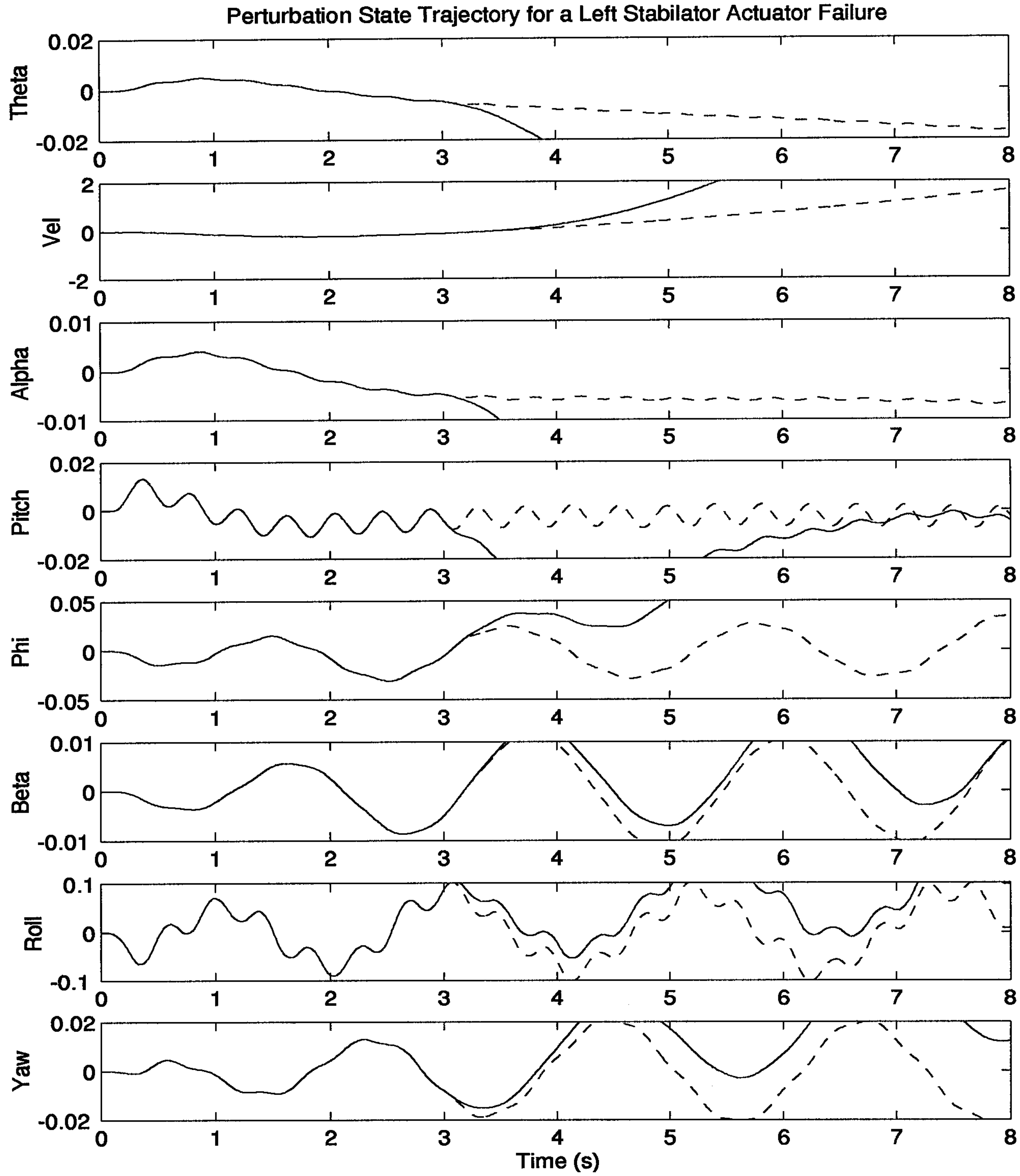


Figure 54. Aircraft State Trajectory: *Left Stabilator* Actuator Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory)

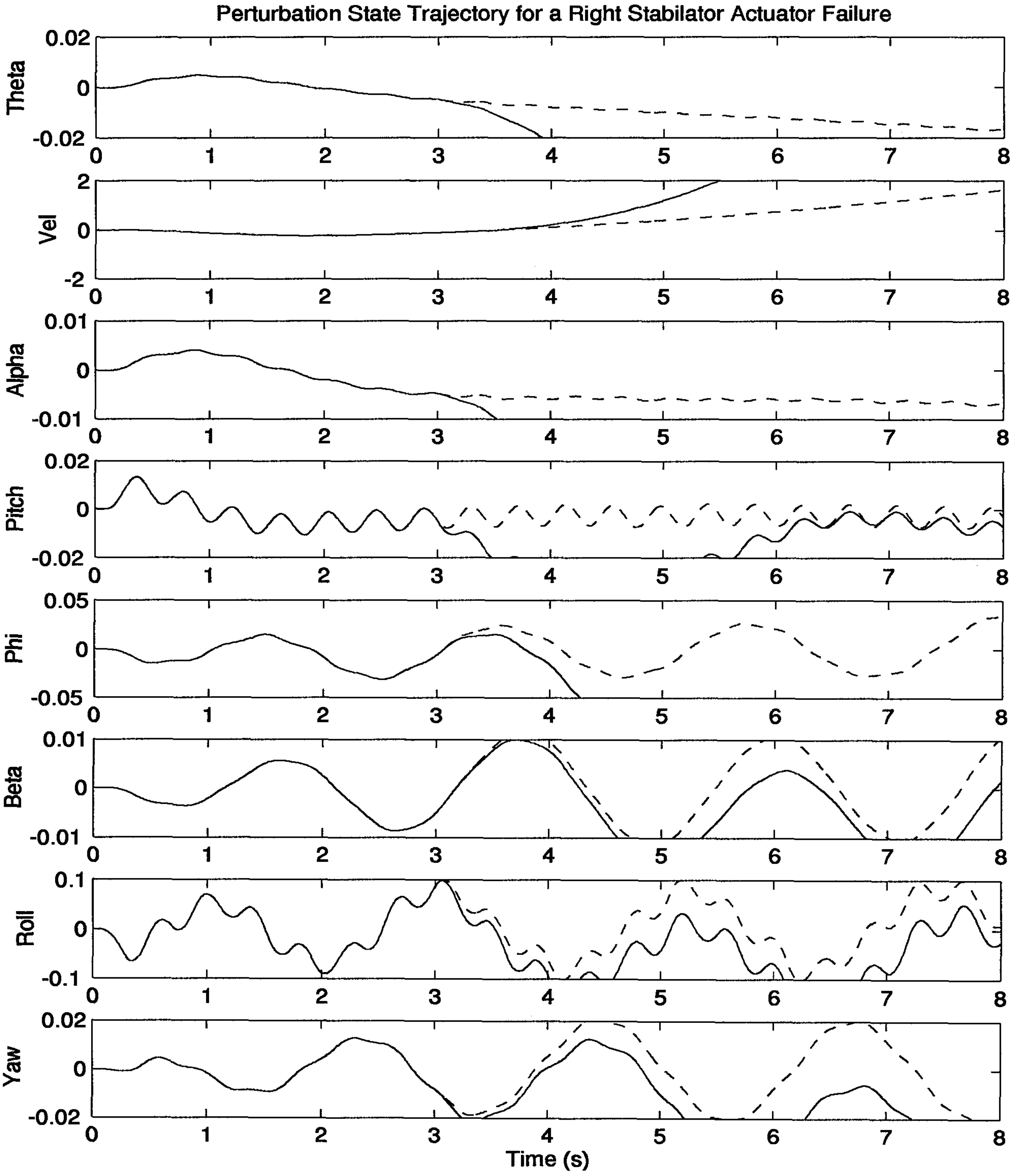


Figure 55. Aircraft State Trajectory: *Right Stabilator* Actuator Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory)

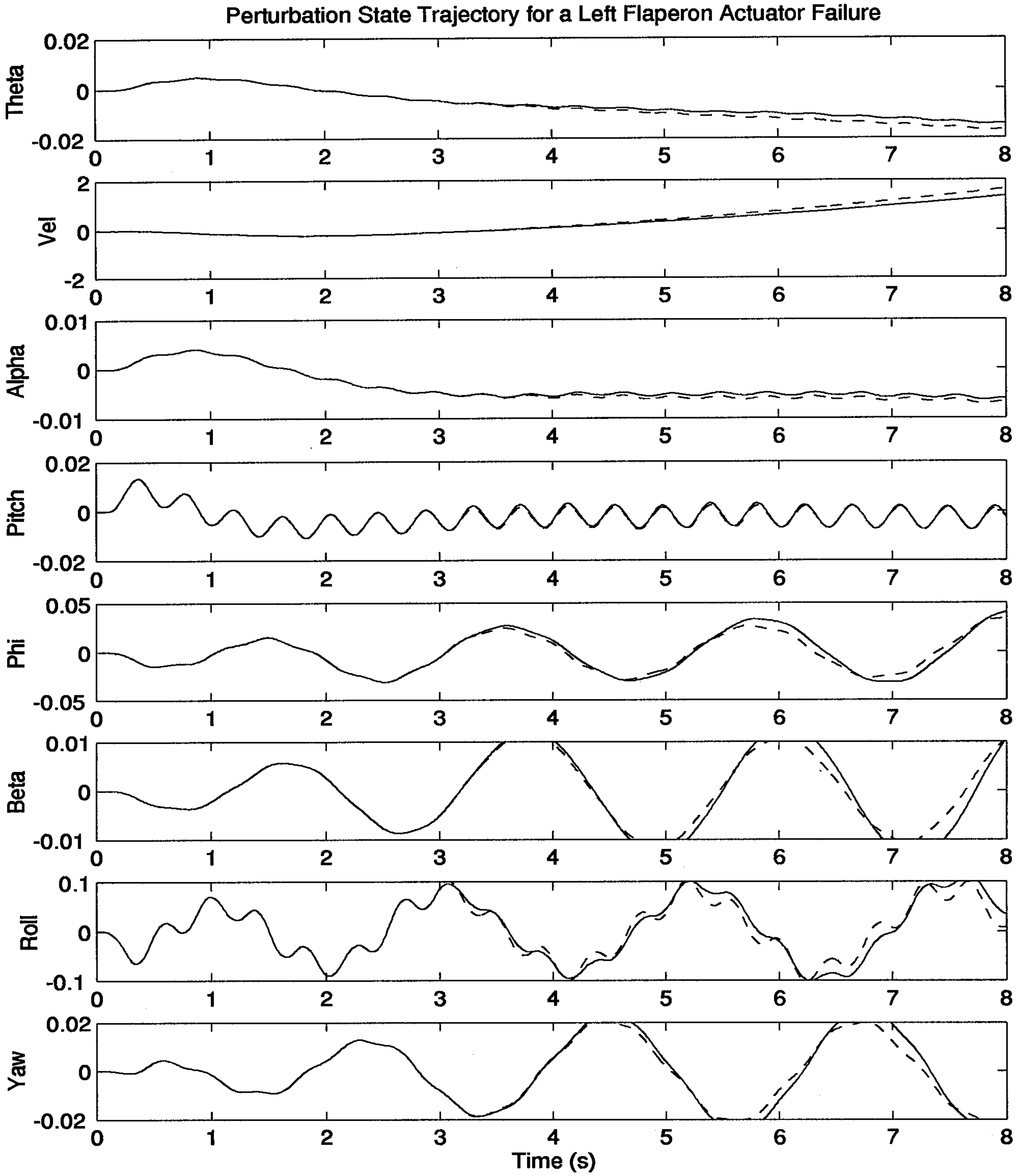


Figure 56. Aircraft State Trajectory: *Left Flaperon* Actuator Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory)

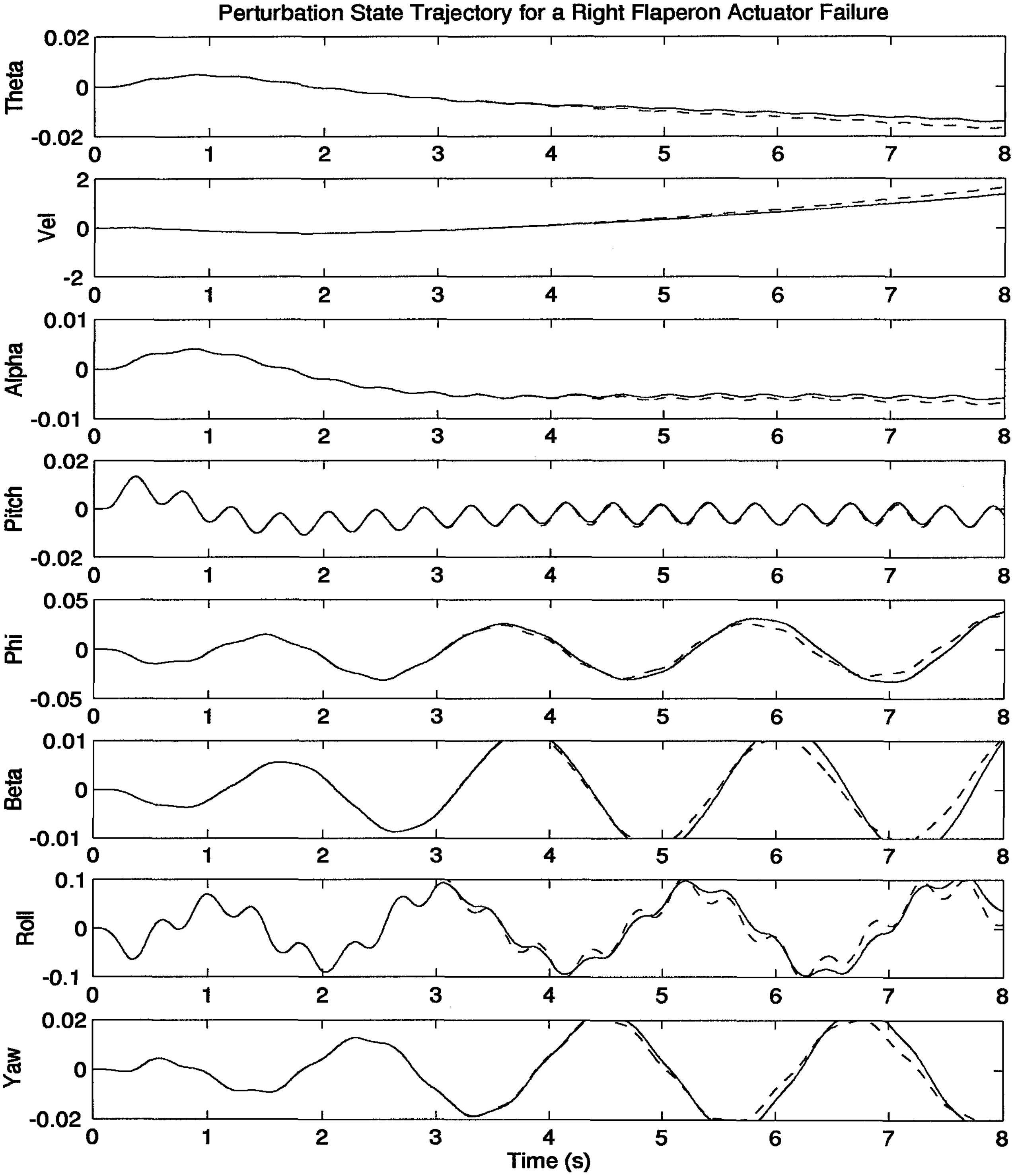


Figure 57. Aircraft State Trajectory: *Right Flaperon* Actuator Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory)

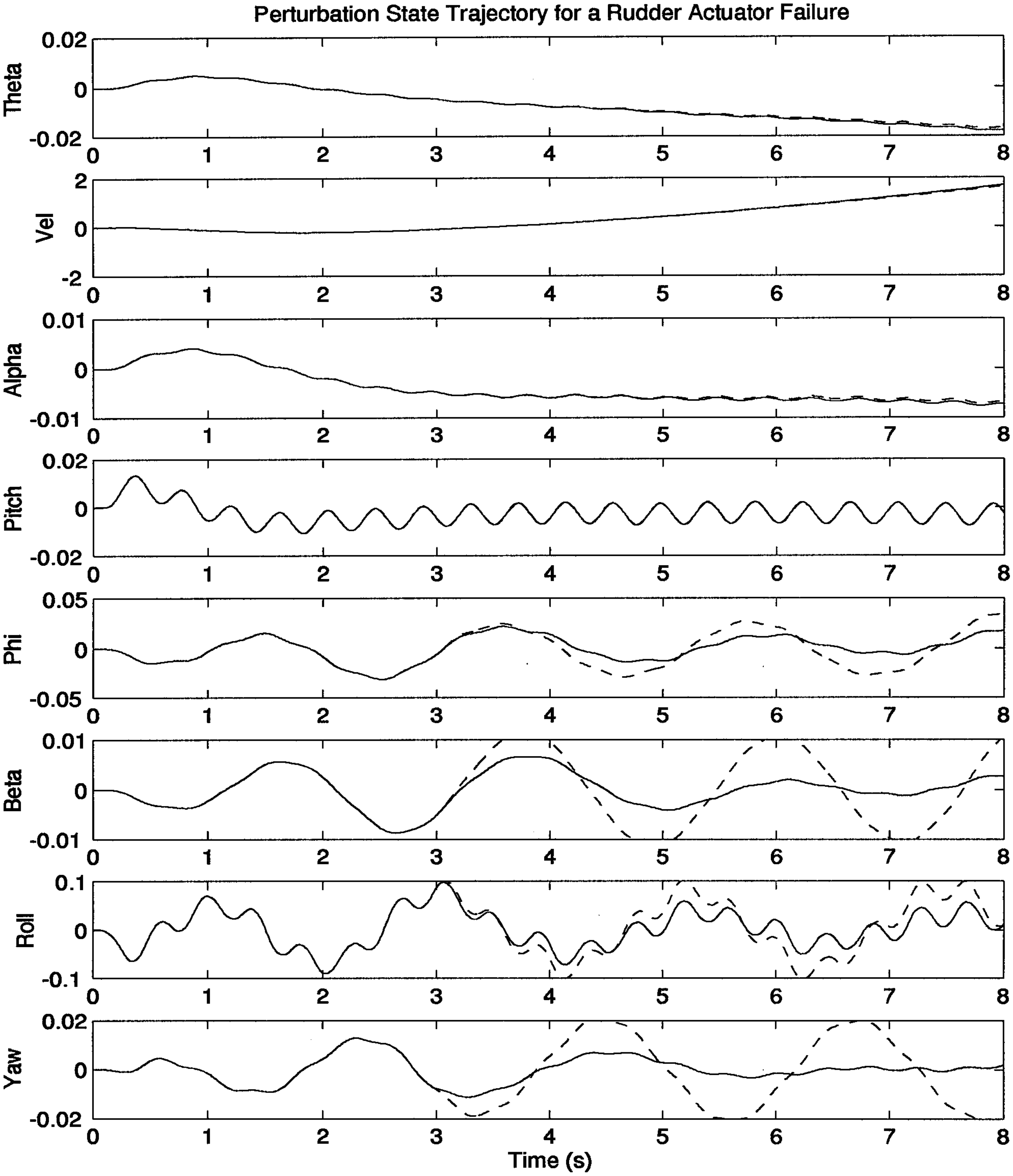


Figure 58. Aircraft State Trajectory: *Rudder* Actuator Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory)

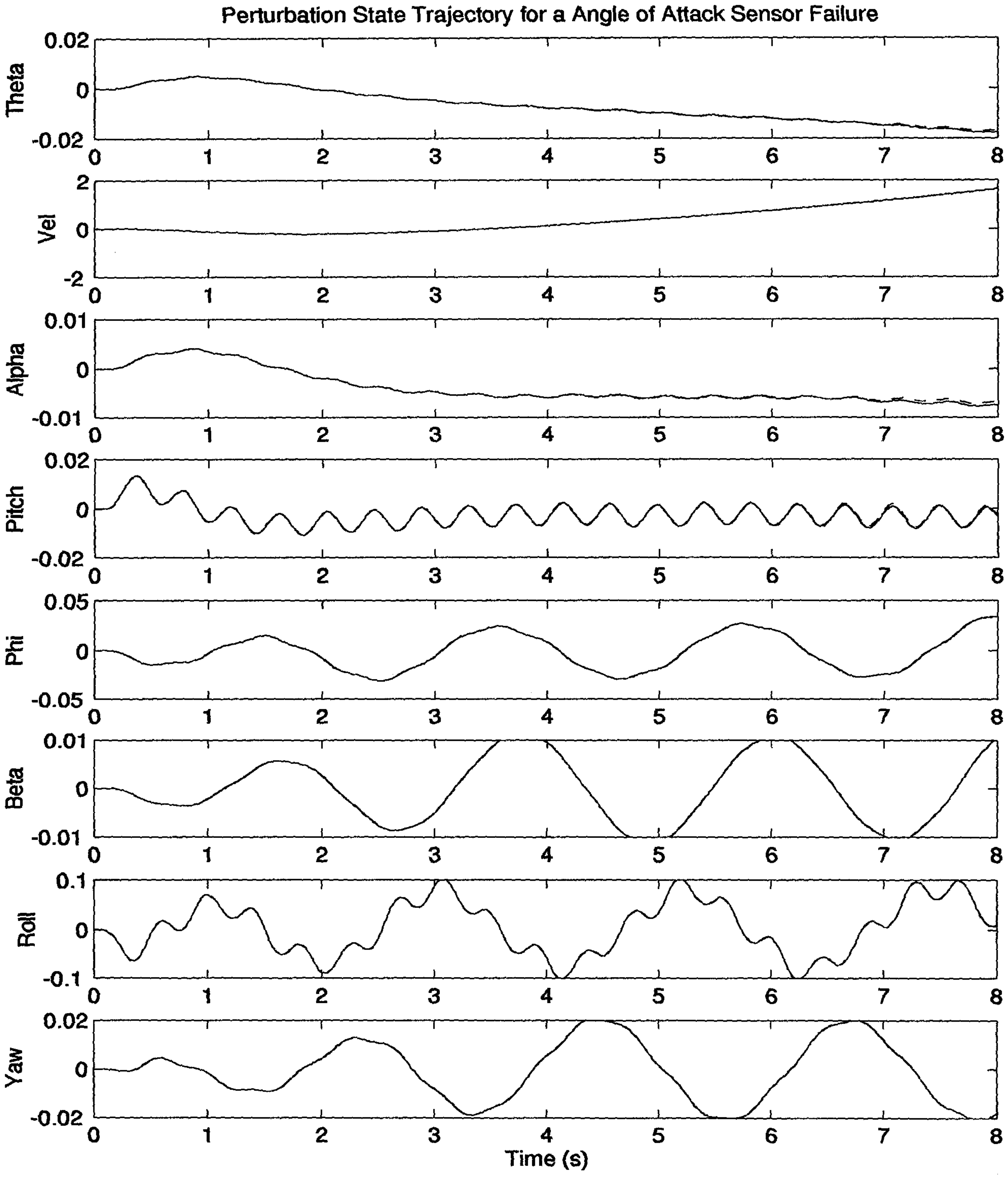


Figure 59. Aircraft State Trajectory: *Angle of Attack* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory)

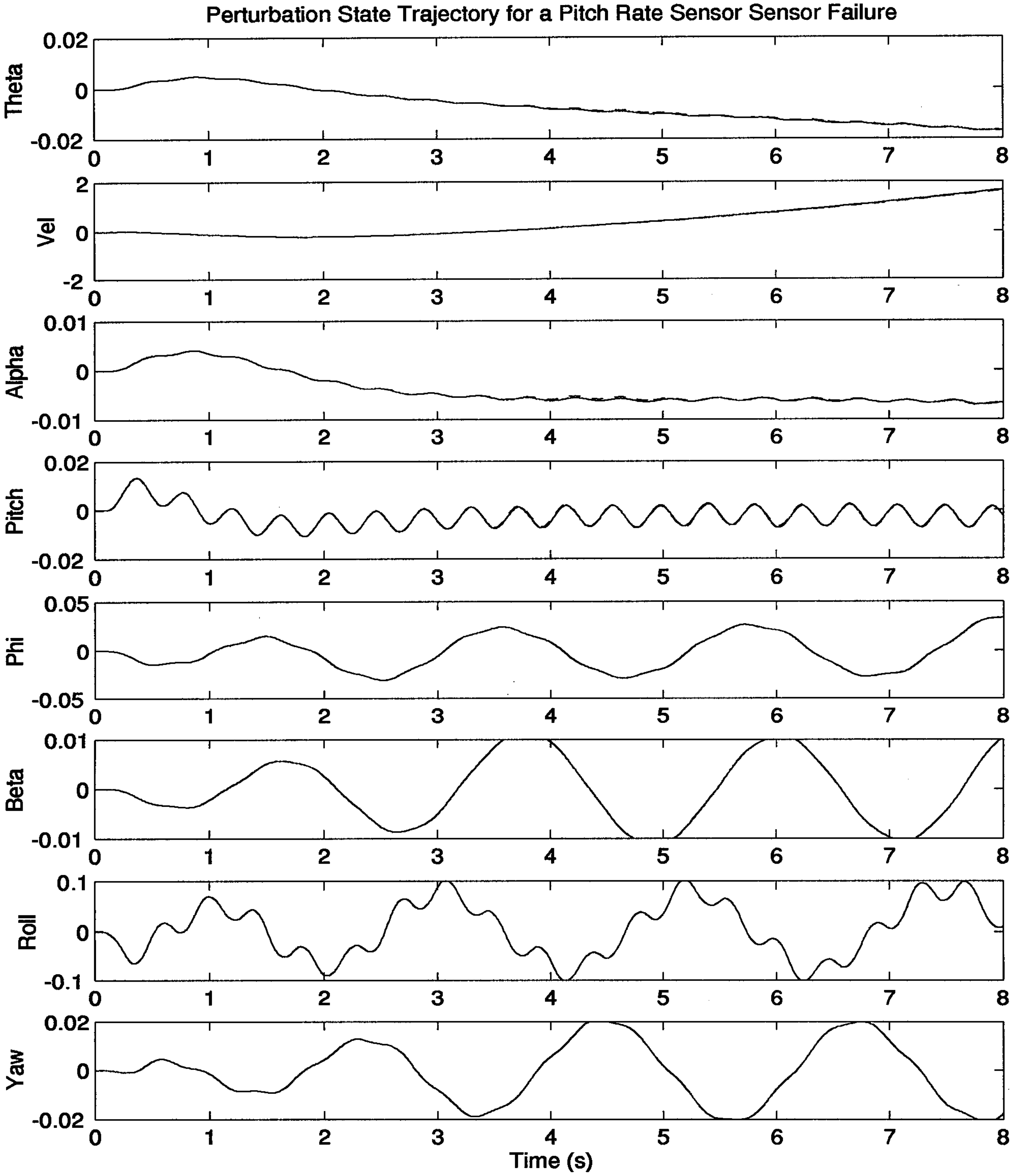


Figure 60. Aircraft State Trajectory: *Pitch Rate* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory)

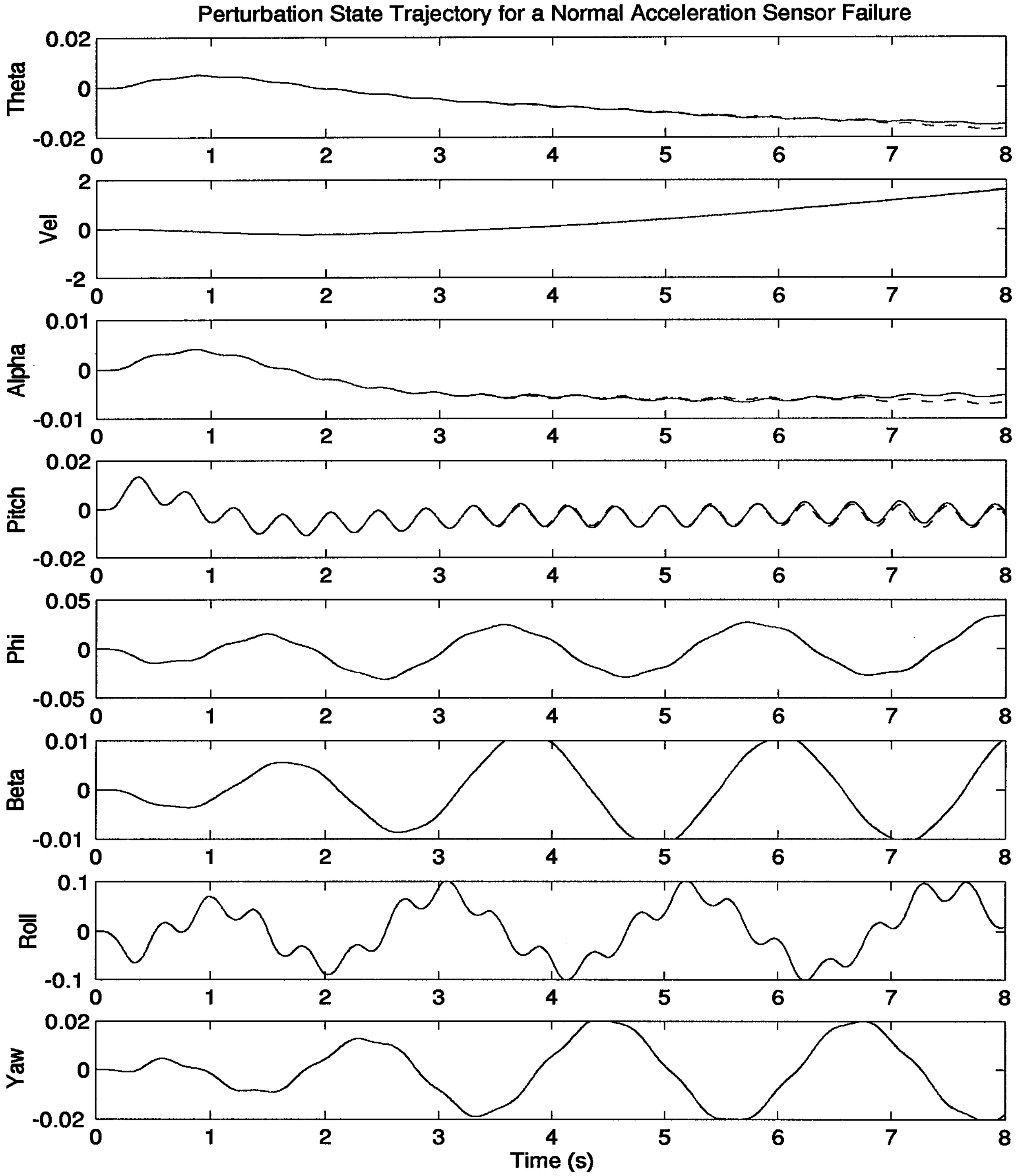


Figure 61. Aircraft State Trajectory: *Normal Acceleration* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory)

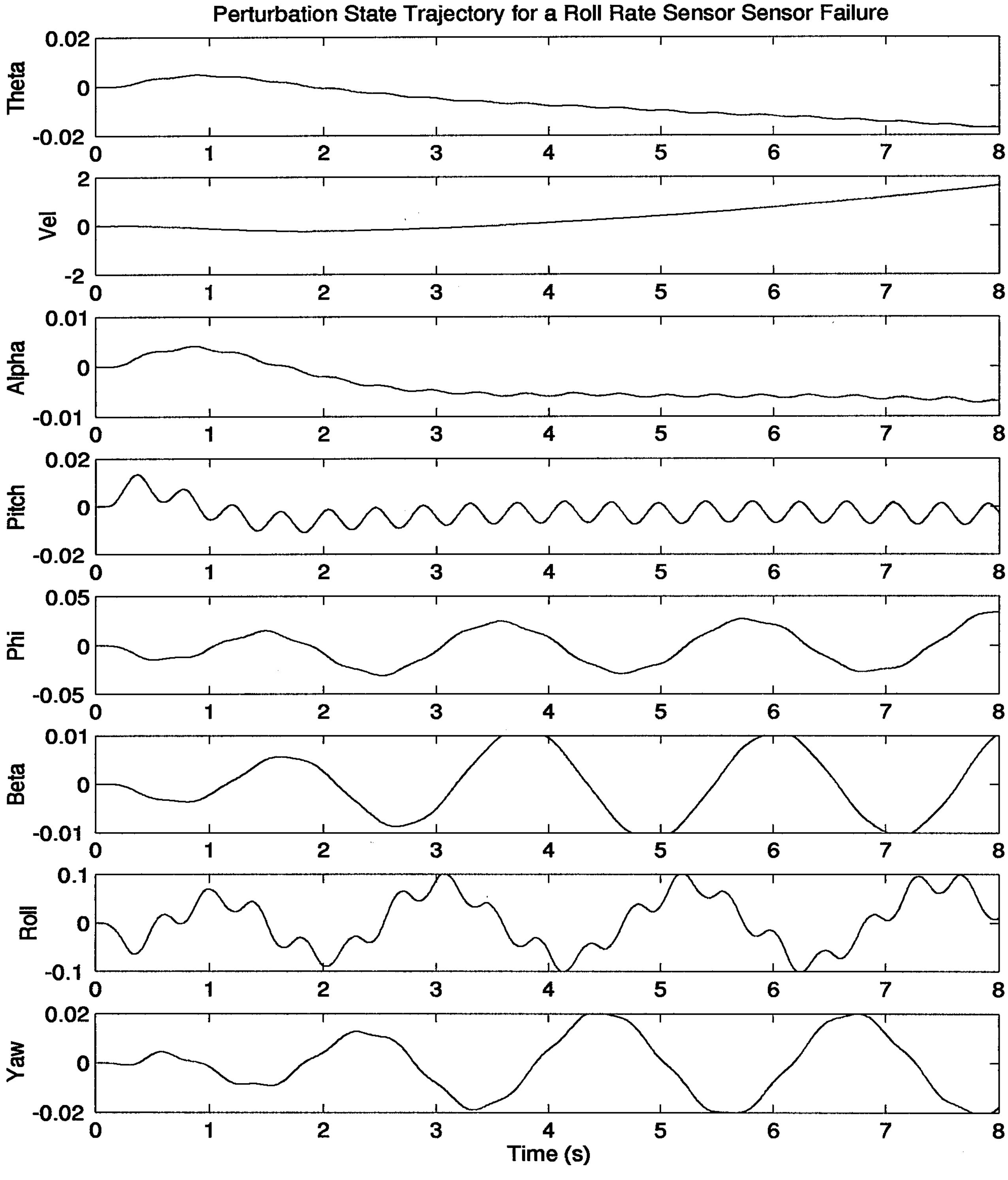


Figure 62. Aircraft State Trajectory: *Roll Rate* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory)

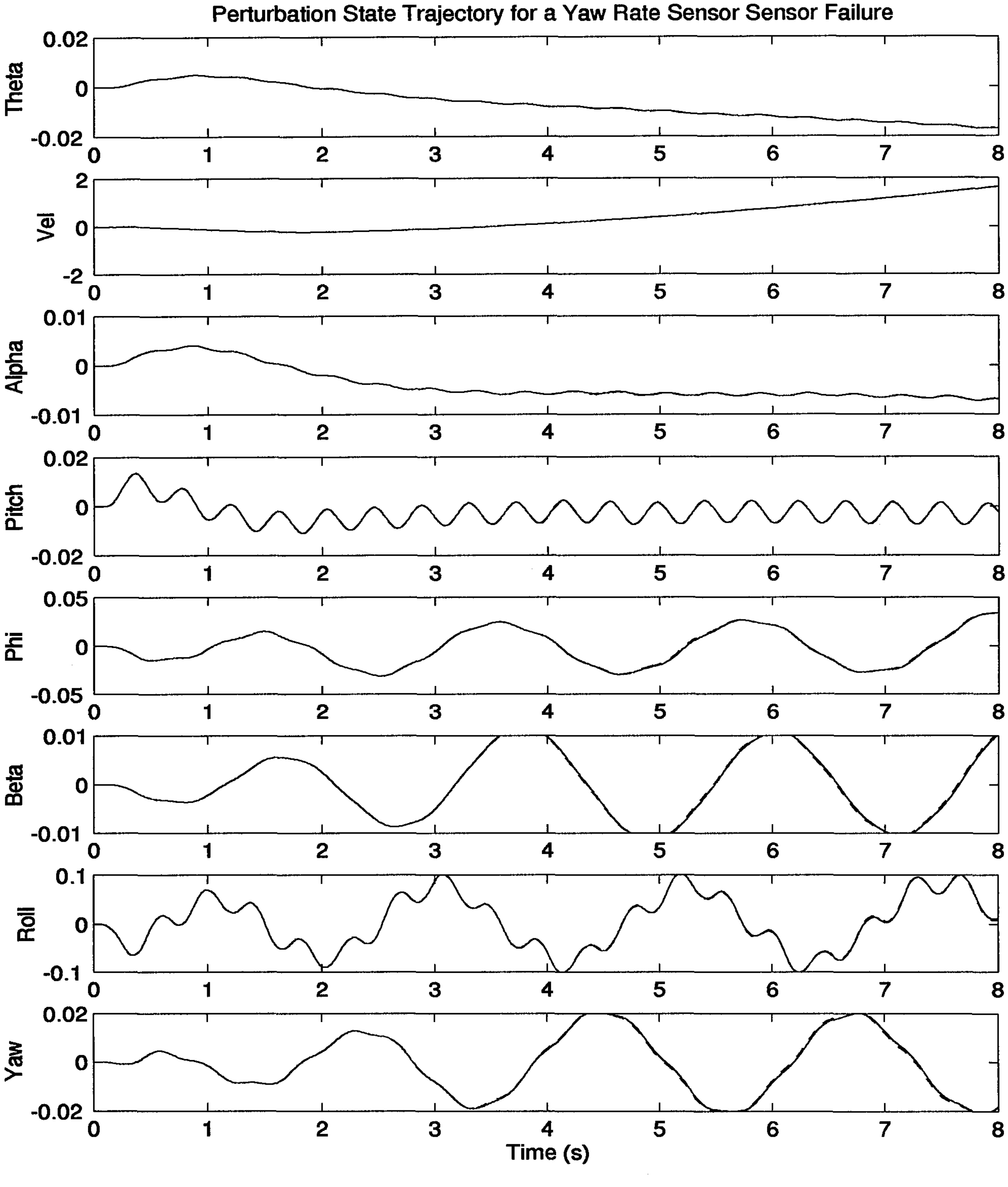


Figure 63. Aircraft State Trajectory: *Yaw Rate* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory)

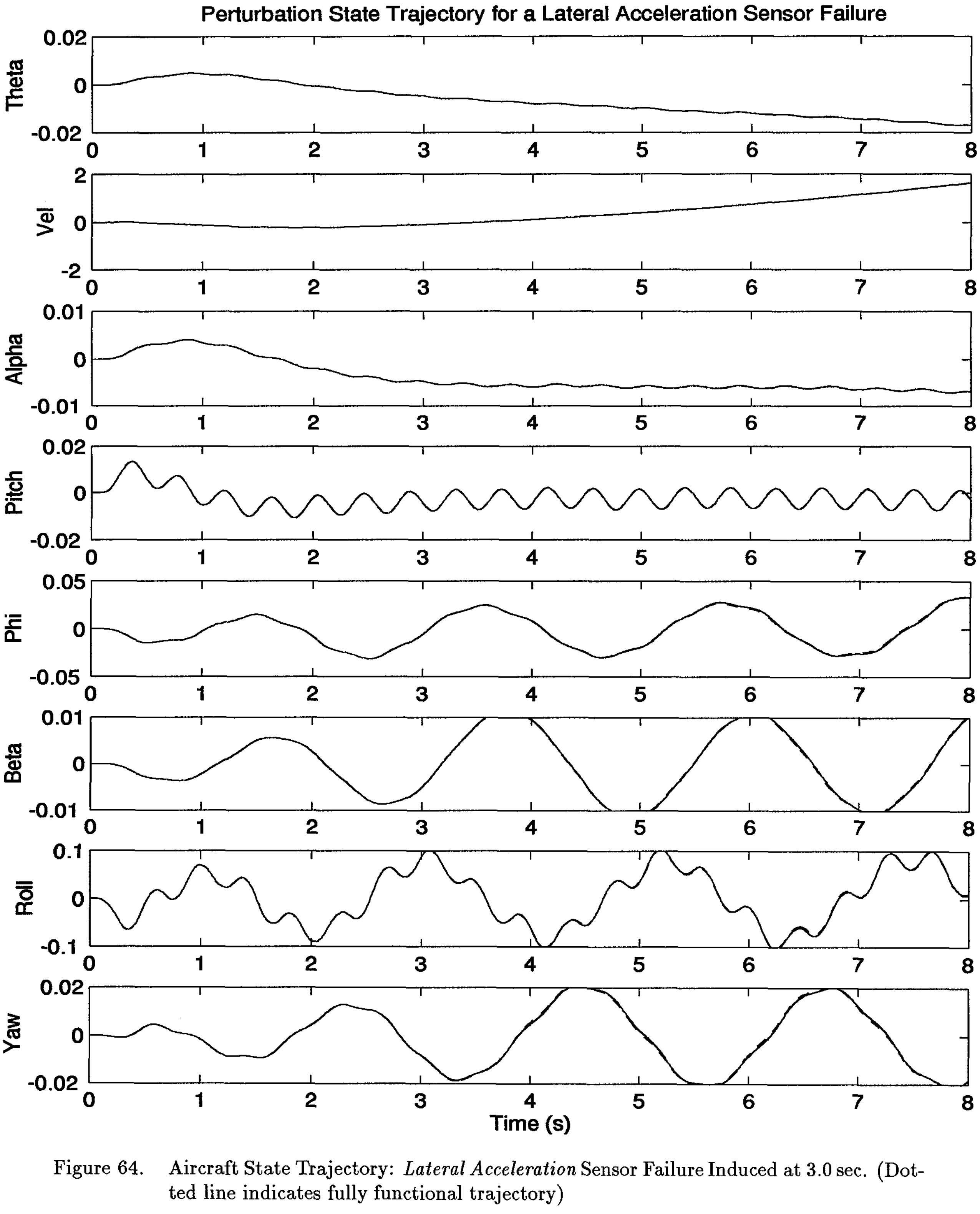


Figure 64. Aircraft State Trajectory: *Lateral Acceleration* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory)

To demonstrate the point further, Figures 65 to 70 depict the same single sensor failure trajectories, only in this case, the MMAE-reconstructed estimates are **not** being supplied to the FCS. The resulting departures speak for themselves.

Since the Kalman filters generate an estimation of the state vector, our MMAE-based control algorithm already provides for flight control robustness to sensor failures. This demonstration further motivates pursuit of MMAC algorithms which include robustness to actuator failures as well.

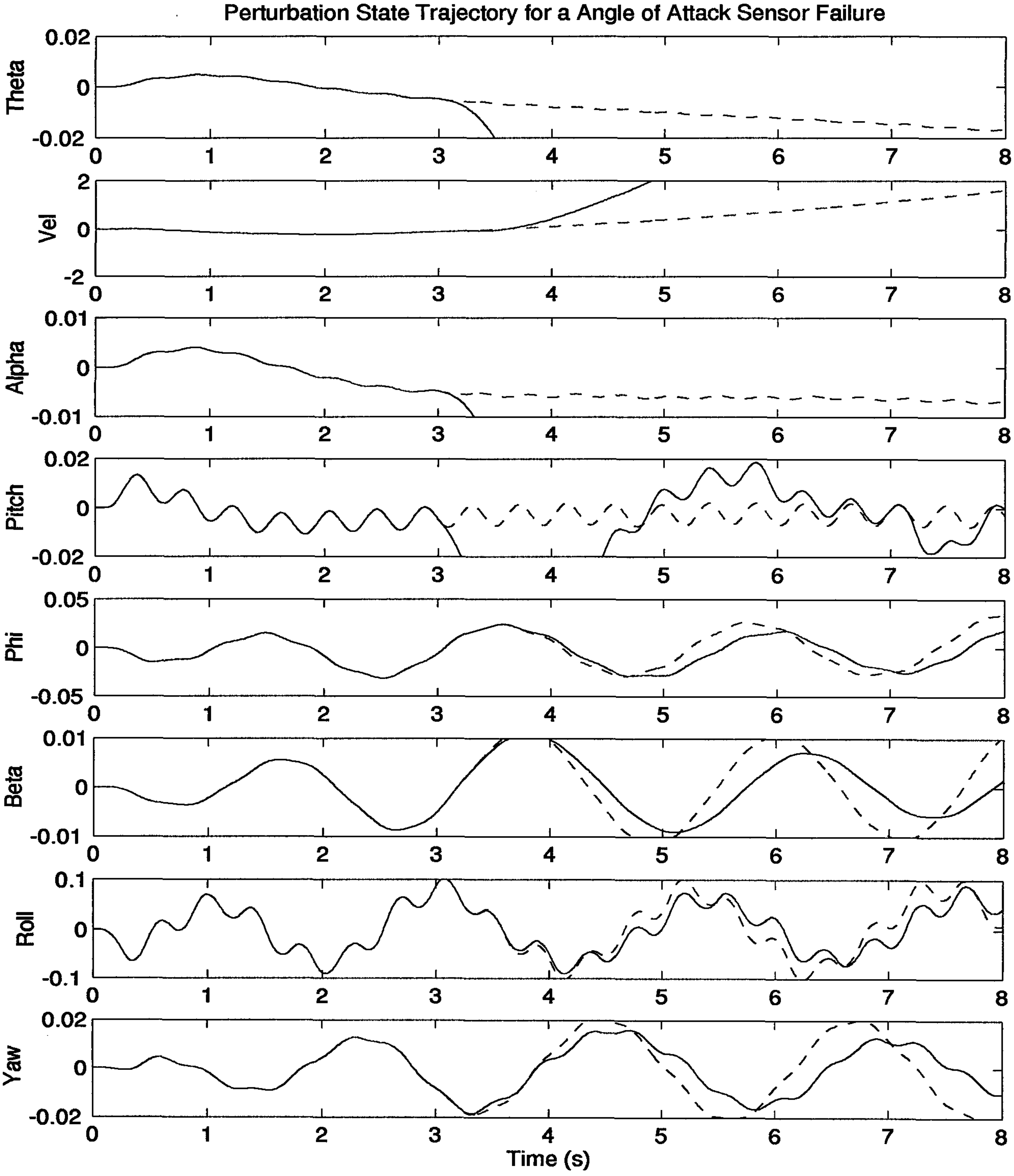


Figure 65. Aircraft State Trajectory: *Angle of Attack* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) **No MMAE Input**

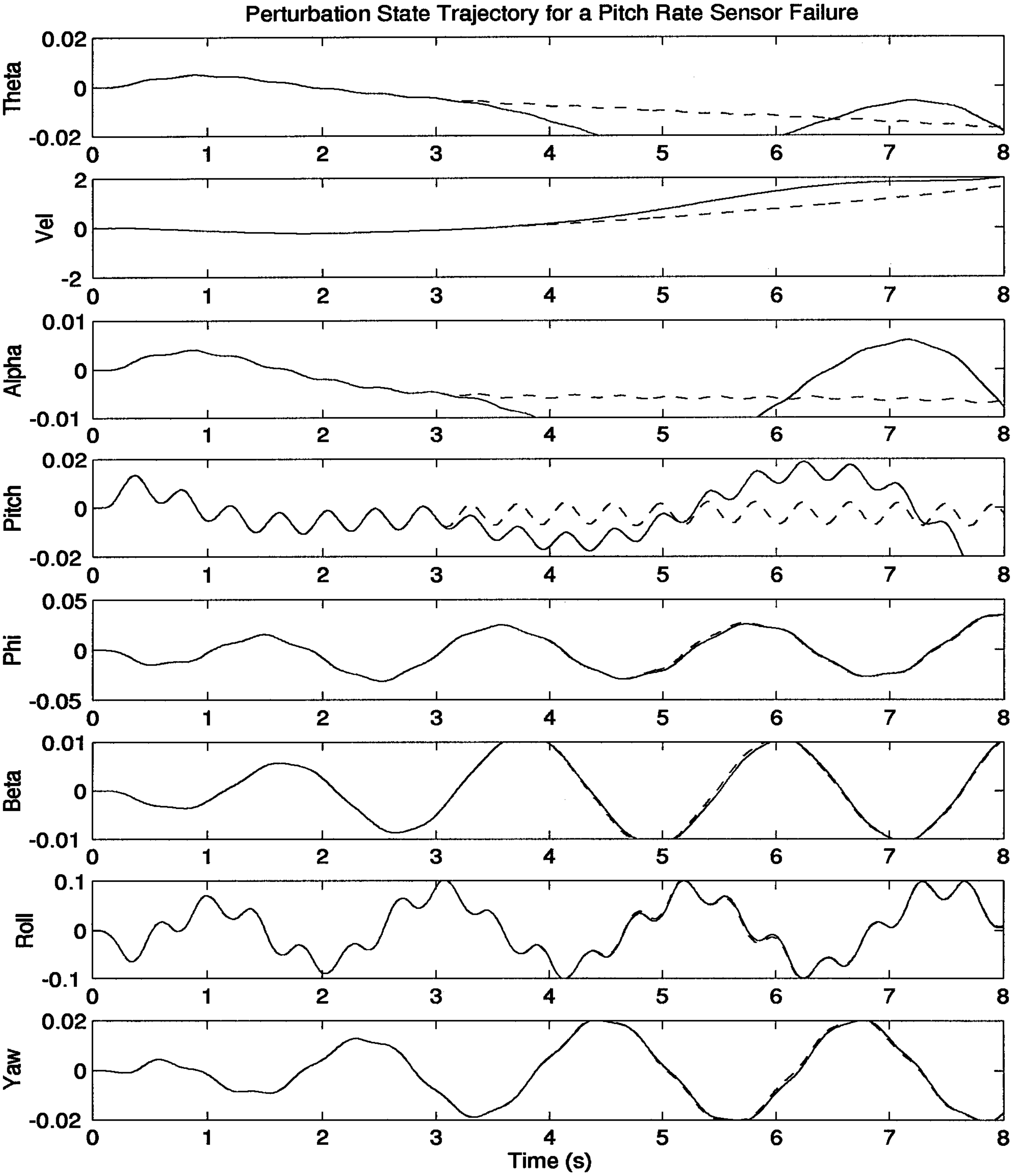


Figure 66. Aircraft State Trajectory: *Pitch Rate* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) **No MMAE Input**

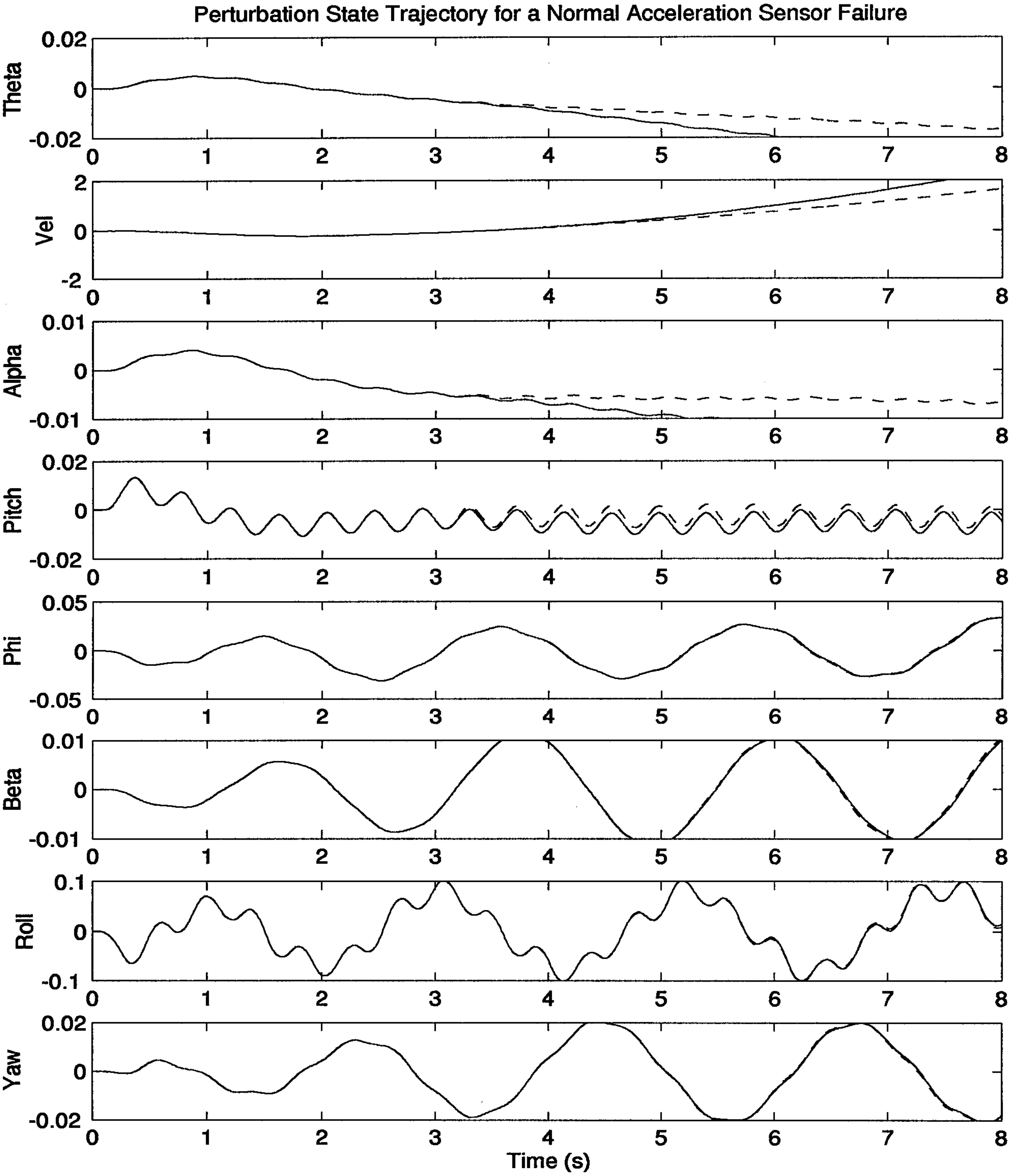


Figure 67. Aircraft State Trajectory: *Normal Acceleration* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) **No MMAE Input**

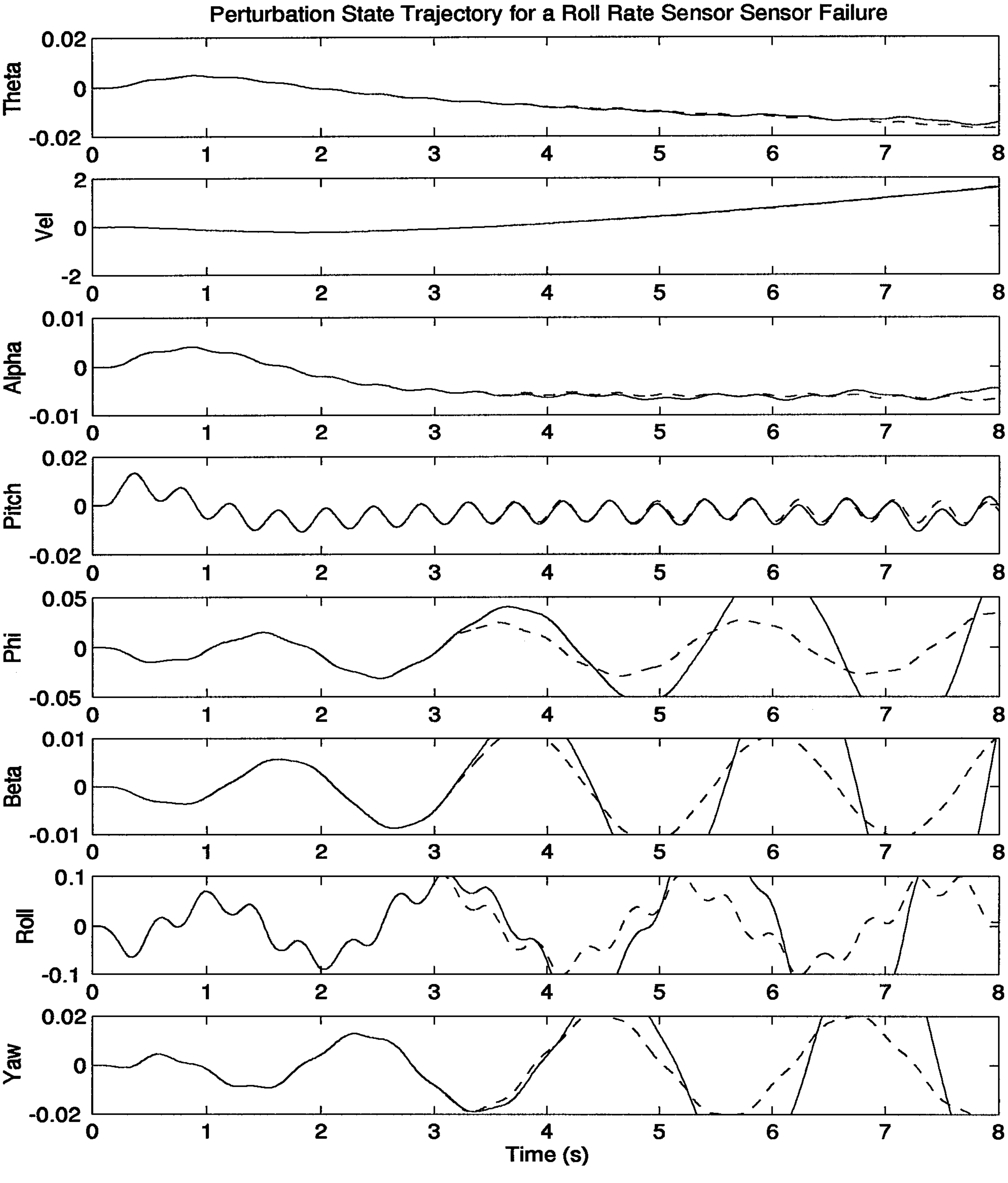


Figure 68. Aircraft State Trajectory: *Roll Rate* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) **No MMAE Input**

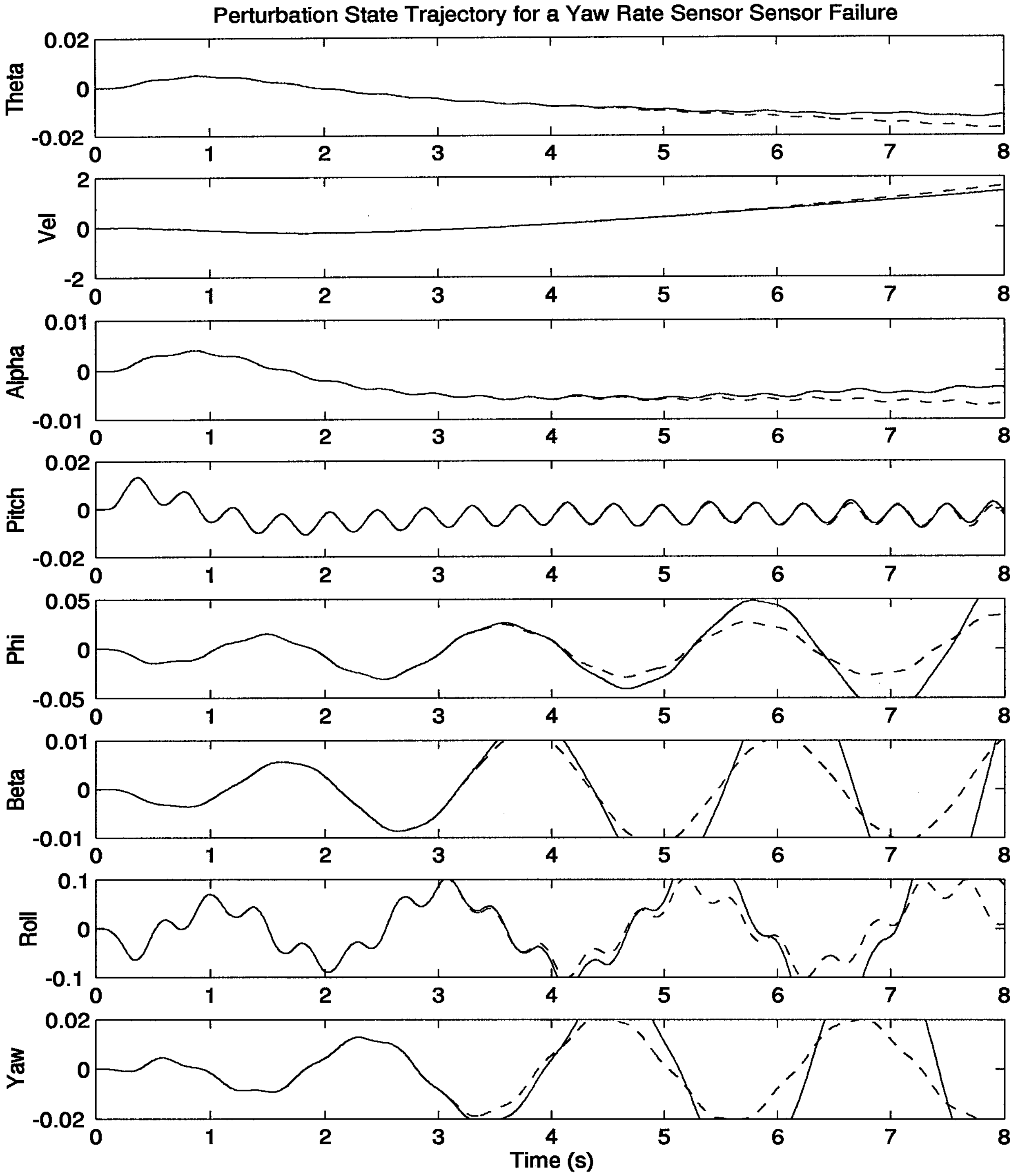


Figure 69. Aircraft State Trajectory: *Yaw Rate* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) **No MMAE Input**

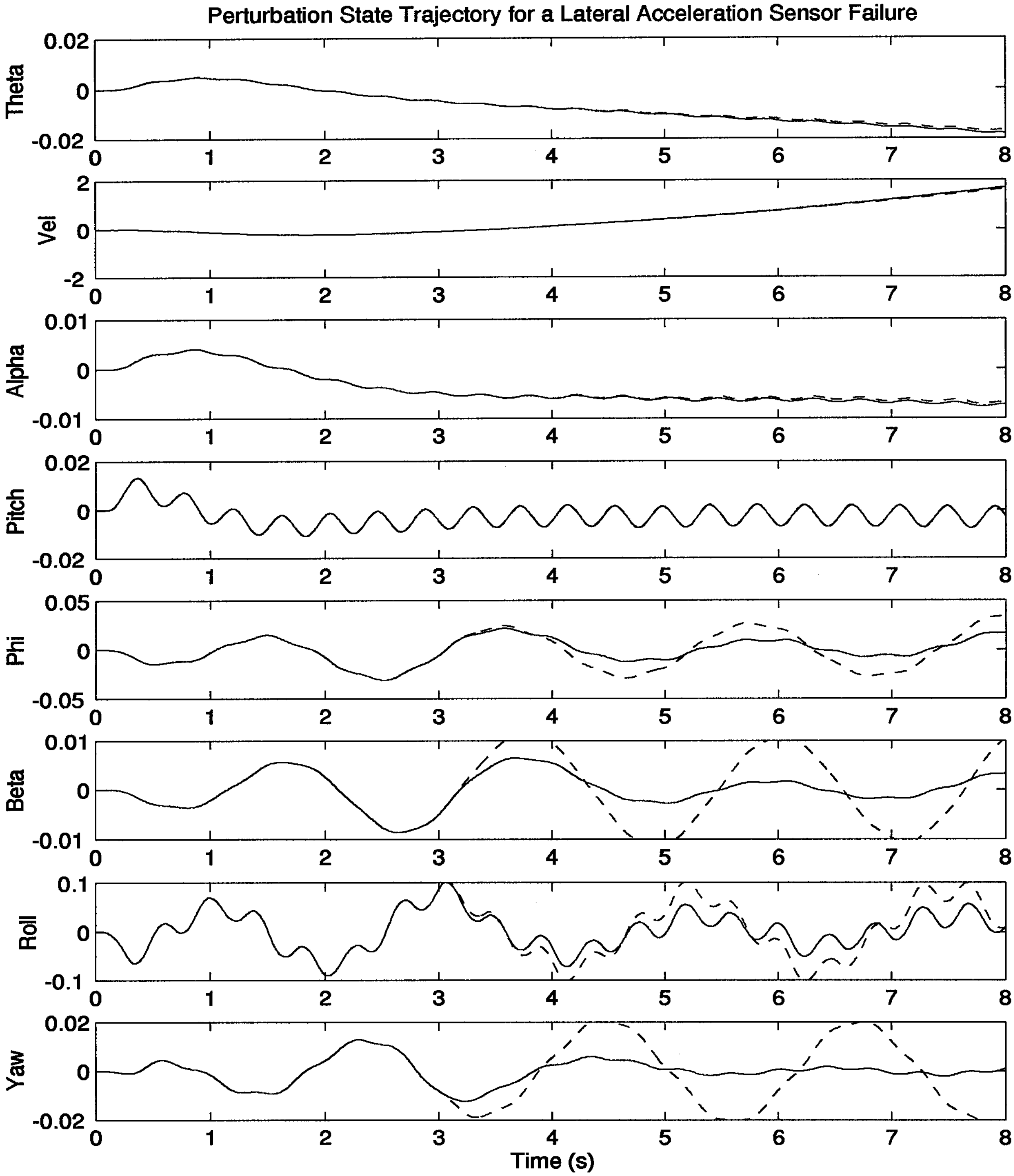


Figure 70. Aircraft State Trajectory: *Lateral Acceleration* Sensor Failure Induced at 3.0 sec. (Dotted line indicates fully functional trajectory) **No MMAE Input**

4.5 Flight Envelope Research

Our attempts at moving toward full-envelope coverage were limited to observing performance off nominal. In other words, we moved the simulation to different points within the F-16 flight envelope, but local to the nominal design point, to observe and characterize the inevitable performance degradation. Recall that our entire filter design is based upon equations of motion linearized for a given trim position in the flight envelope. Operation of the algorithm, therefore, at a point for which it was not specifically designed, implies inherent model mismatch. The question then is, how much mismatch can the algorithm tolerate before performance degrades to unacceptable levels.

To answer this question, we began with the simulation and filter design of the single-failure scenario. Points were selected away from nominal, along axial Mach and altitude directions. When unacceptable performance was reached in each of the four directions, the local neighborhood was established. Four additional points were later added to check extreme effects outside this local envelope. Table 10 shows the points selected within the envelope (points 1 through 20) as well as the additional extrema (points 21 through 24). At each of the points shown in Table 10, we simulated the entire single-failure scenario. From these results, we hoped to assess the algorithm's robustness off nominal.

4.5.1 Characterization of Flight Envelope Results. Before getting to the plots, a couple of important characterizations can already be made. In preparing for the simulations, we learned three very important points: accurate trim variables are critical, false alarms are the limiting factor, and the algorithm is **not** very robust off nominal. The following paragraphs address these issues in more detail.

First, recalling that our design is a perturbation model, trim values of input variables are important since they must be subtracted off all incoming data to produce a perturbation variable. During research in the single-failure scenario, we assumed *a priori* knowledge of the variables which were non-zero at nominal. Incidentally, they were forward velocity, angle of attack, normal acceler-

	0.350	…	0.380	0.385	0.390	0.395	0.400	0.405	0.410	…	0.450
25000							21				
⋮											
22000					1		2				
21500				3		4		5			
21000			6		7		8		9		
20500				10		11		12			
20000	22		13		14		⋆		15		24
19500				16		17		18			
19000					19		20				
⋮											
15000							23				

Table 10. Envelope Investigation Matrix: Altitude (row) vs Mach (column), ⋆ denotes nominal point

ation (very small), flaperon deflection, and stabilator deflection. The problem was, inaccurate trim values resulted in poor performance. It took deep investigation to find that the incoming perturbation variables weren't beginning exactly at zero. Needless to say, this issue extends to the full envelope research. For example, attempting to take the MMAE designed for one point and operate it at another point 10 $\frac{ft}{s}$ faster would yield an initial *perturbation* error of 10 $\frac{ft}{s}$. In perturbation scales, this is a very significant departure and indeed, no adequate performance could be achieved off nominal without resetting the appropriate nominal trim variables. We therefore must consider possible ways of resetting these variables in real-time usage. One might be to gain-schedule the trims. Another would be to incorporate sensor data available on the aircraft somehow (additional sensors might be required). In this research, we used the SRF simulation to determine appropriate trim values for each point within the envelope.

A second early characterization is that false alarms are probably going to be the limiting factor in off-nominal robustness. It has been our experience throughout this research that, even when false alarms were being problematic, induced failures continued to be consistently identified. It seems the hard failures' effects dominate over whatever confusion the MMAE was having in the presence of no failures. An example of this was discussed briefly in the dual failure scenario results section. Even though the stabilators consistently false-alarm each other, the second (induced) failure effects dominated and led to successful identification. We therefore felt that, before we got far enough away from nominal in the flight envelope for low dynamic pressure (causing *missed* alarms) to be a factor, false alarms due to model mismatch would define our local envelope. In fact, that is exactly what happened. We expect this characteristic to be true throughout the envelope as well.

Lastly, one needs only to look at the admissible local envelope in Table 10 to see that the algorithm is not very robust off nominal. The range from 0.38 to 0.41 Mach is only about 30 $\frac{ft}{s}$, less than 2% of the flight envelope. Altitude variability fares better, 19000 to 22000 ft, but still represents only approximately 6% of the envelope. These results alone argue rather convincingly for a gain-scheduling scheme accomplished interpolatively for each of the elemental filters within the MMAE structure. Too many individual filters would have to be designed for any type of handoff scheme among discrete point designs as suggested in Figure 3, Chapter 1. Even with these characterizations known to some certainty ahead of time, we proceeded with the investigation to see what exactly does happen off-nominal within this locally defined envelope.

Figures 71 to 78 display the results of our investigation. Each corresponds to a radial direction away from the nominal. Within the plots are multiple sets of single failure performance data corresponding to each envelope point crossed along that radial line. For example, Figure 71 displays all of the data generated at the nominal and envelope points 8, 2, and 21. This is the radial direction vertical from the nominal in Table 10. The desired effect is to make any degradation away from nominal along a radial line more apparent by allowing direct comparison of the results. Returning to the example, each plot in Figure 71 contains four traces. The solid line corresponds to the nominal, the dashed to envelope point 8, the dotted to envelope point 2, and the dot/dash to envelope point 21. Unlike the figures used in past sections, theses contain the entire sets of single failure scenario data. By that, we mean that the **FF** plot corresponds to the average of 10 Monte-Carlo-run-generated **FF** plots when no failure has been induced. In like manner, the **L ST** plot now corresponds to the average of 10-Monte-Carlo-generated **L ST** plots when a left stabilator actuator failure has been induced. Now that the information contained in the plots is clear, we proceed to analyze the results.

At a glance, our hypothesis that false alarms would be the driver are confirmed. Figures 71 to 74 which go out to the extrema show significant dropouts in the fully functional **FF** plot. Remember that these traces are averages and thus hint at false alarms being declared on individual runs. In some cases, performance of the difficult failures such as rudder actuator and pitch rate sensor degrade and in others show improvement as the simulation varies further from nominal conditions. One might expect to see some sort of trend showing faster convergence times and increased false alarms as dynamic pressure increases and slower convergence times and increased missed alarms as dynamic pressure decreases. These plots don't support that conjecture however. While the logic is sound, as mentioned before, we believe that within such a localized envelope, dynamic pressure doesn't have a chance really to become a player. Other factors, undetermined in this research, are at work.

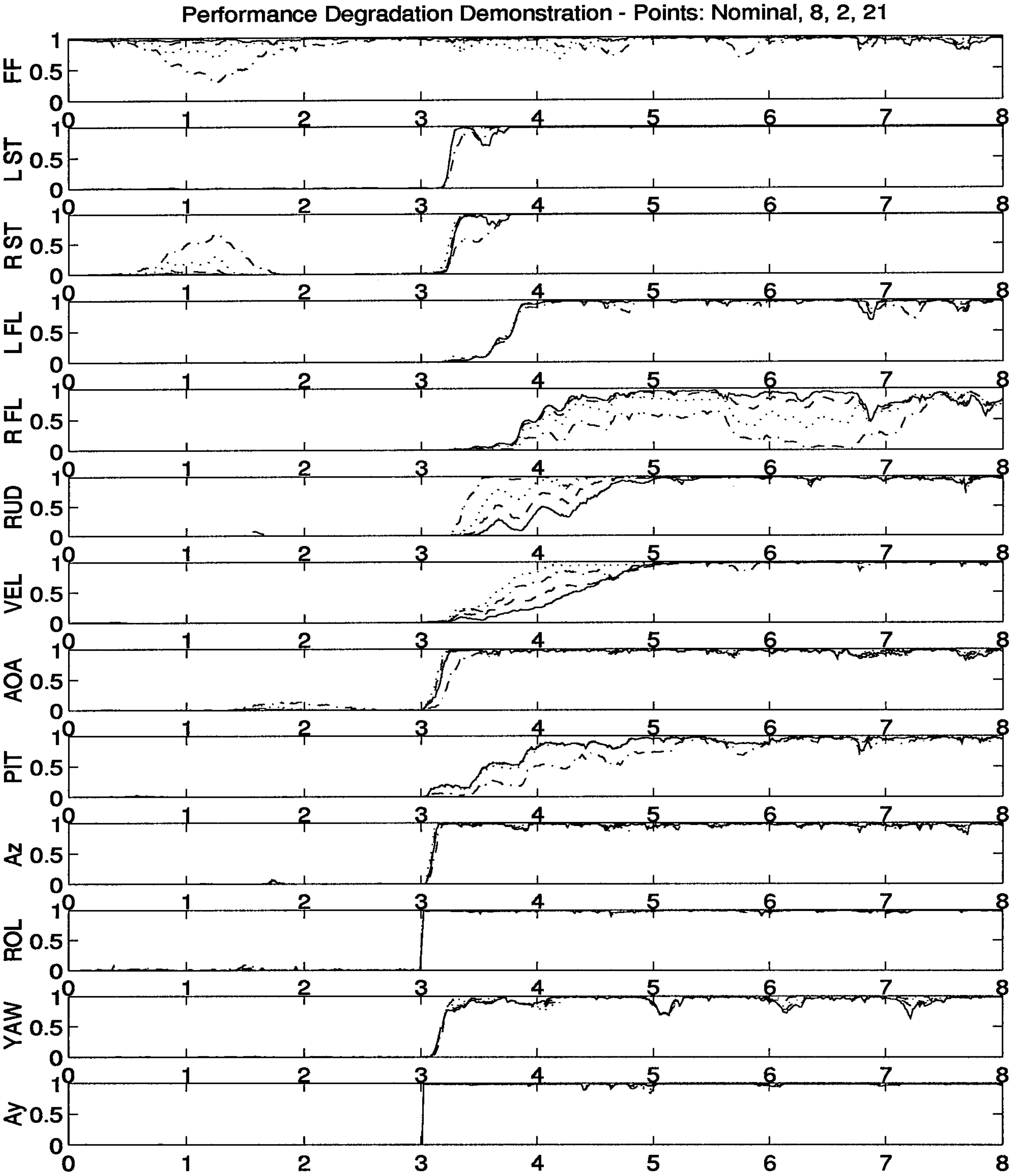


Figure 71. Envelope Radial Line 1: – Nominal, - - Env pt #8, ·· Env pt #2, -·- Env pt #21

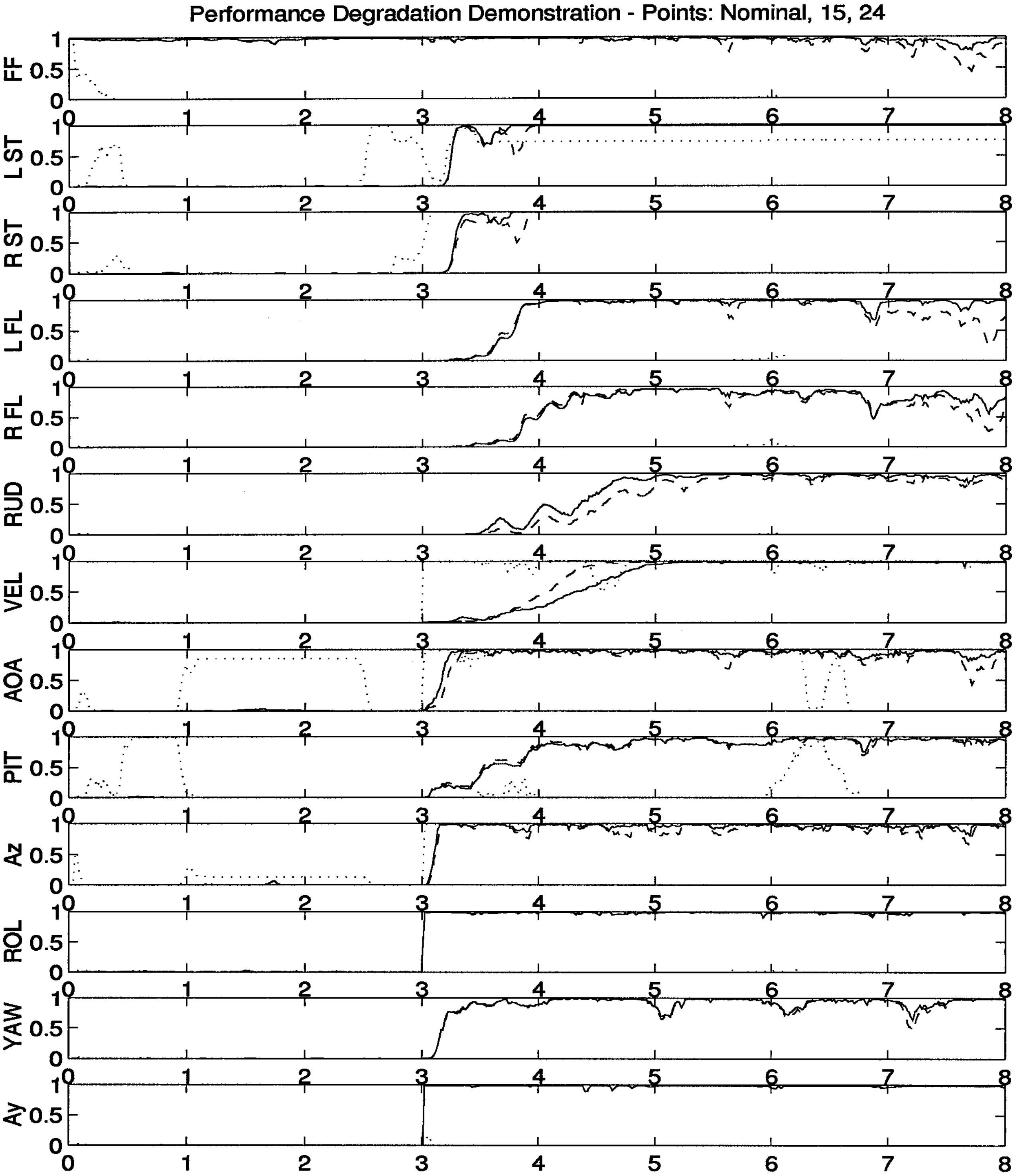


Figure 72. Envelope Radial Line 2: – Nominal, - - Env pt #15, ·· Env pt #24

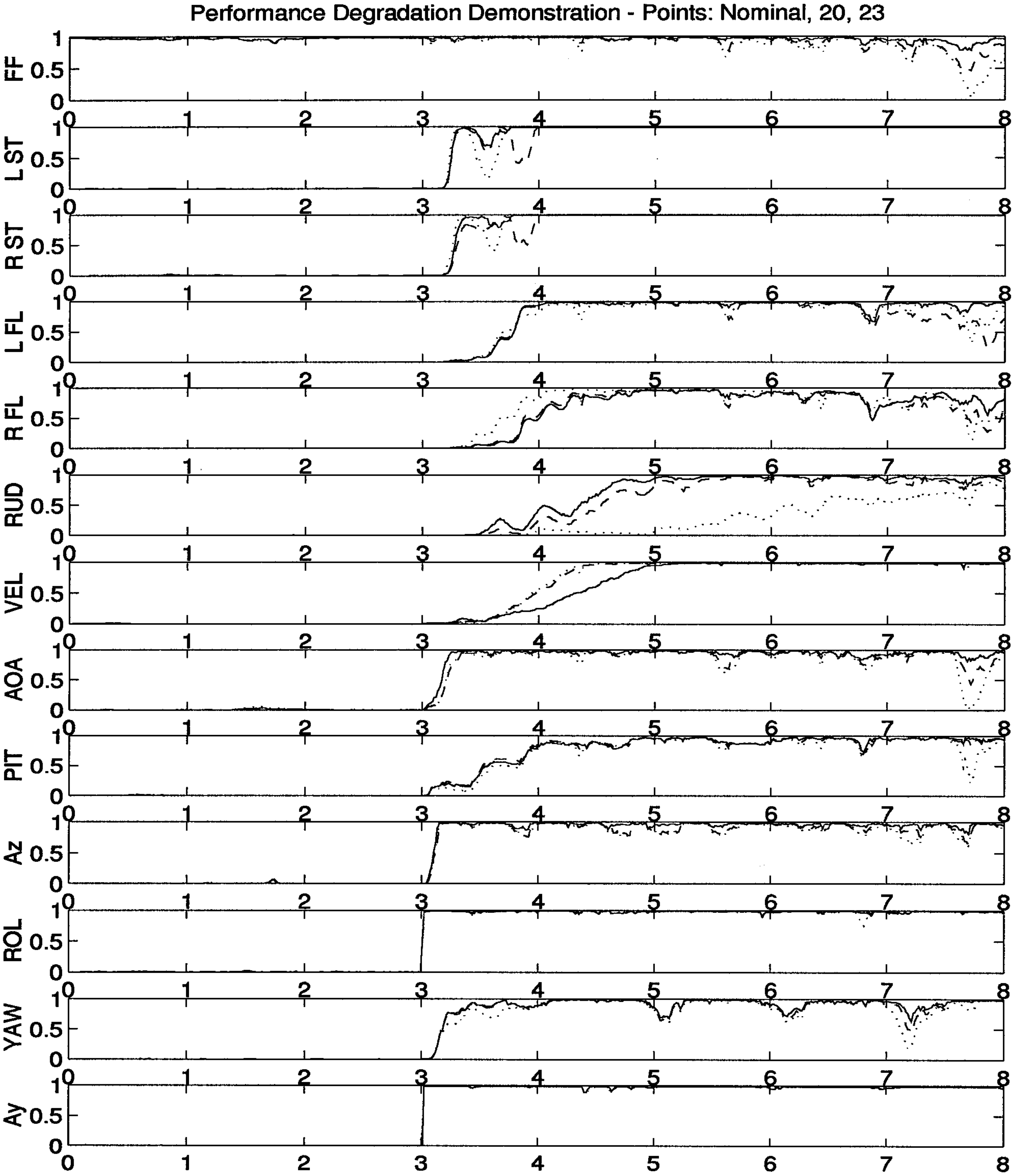


Figure 73. Envelope Radial Line 3: – Nominal, - - Env pt #20, ·· Env pt #23

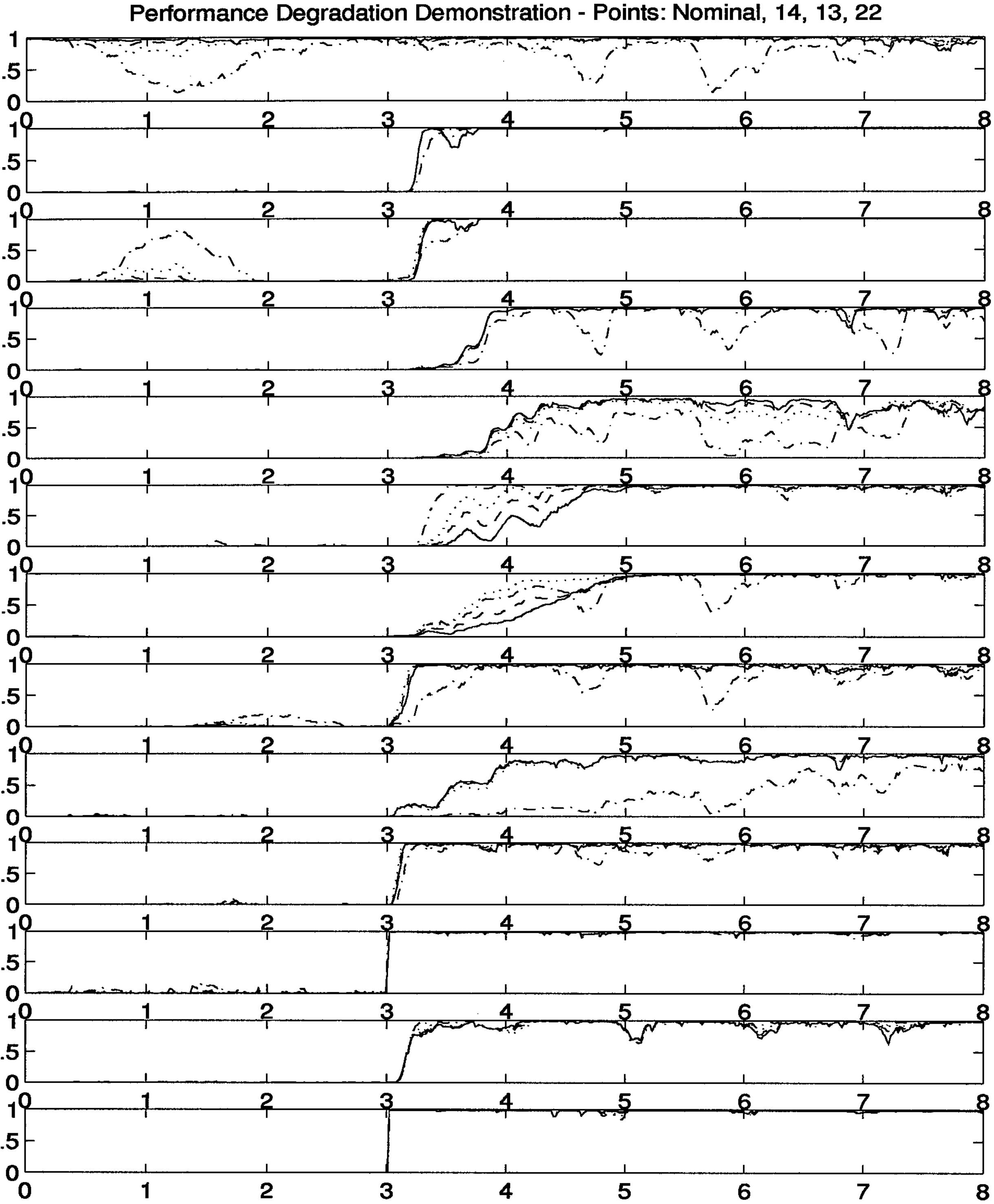


Figure 74. Envelope Radial Line 4: – Nominal, - - Env pt #14, ·· Env pt #13, -·- Env pt #22

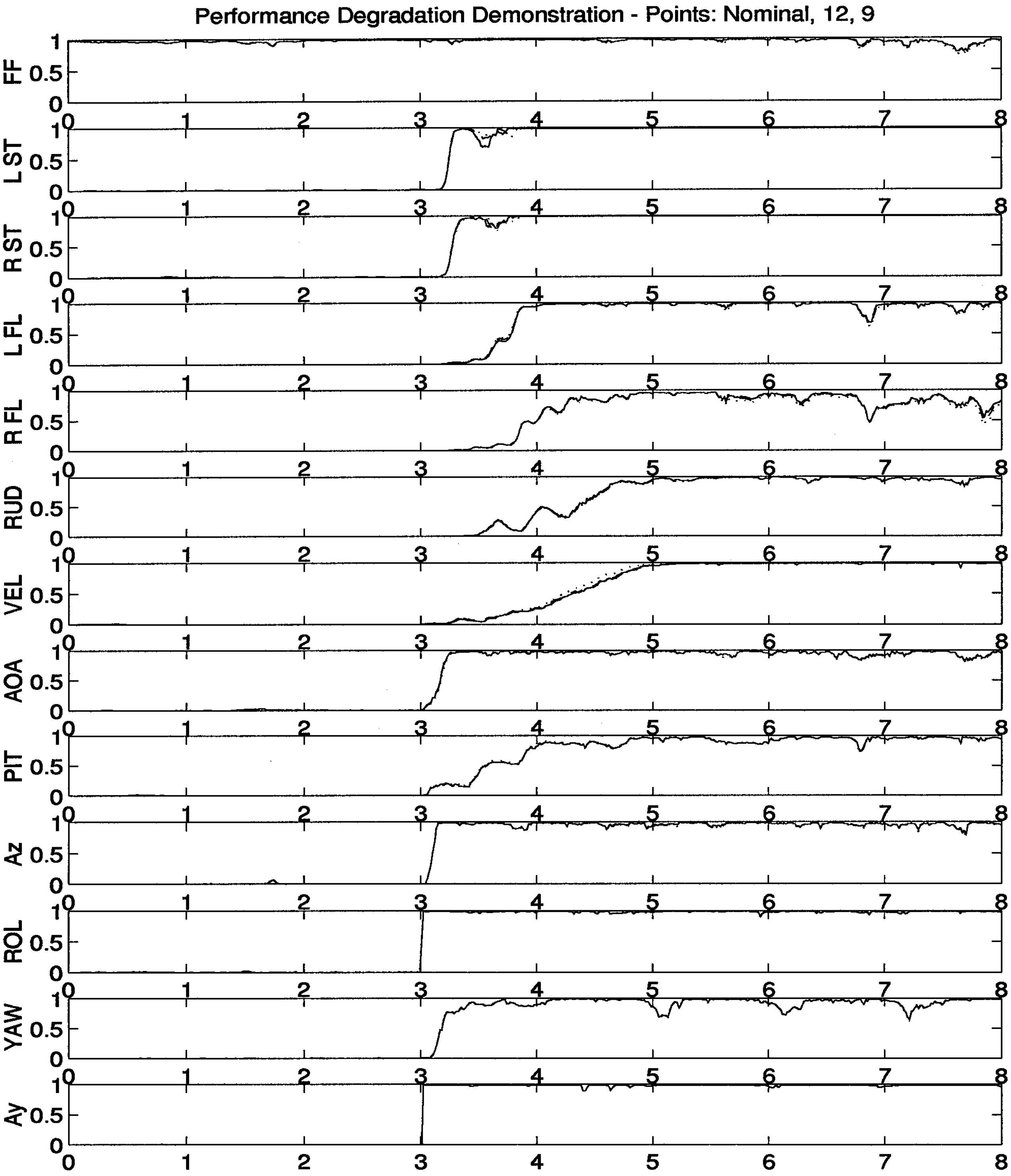


Figure 75. Envelope Radial Line 5: – Nominal, - - Env pt #12, ·· Env pt #9

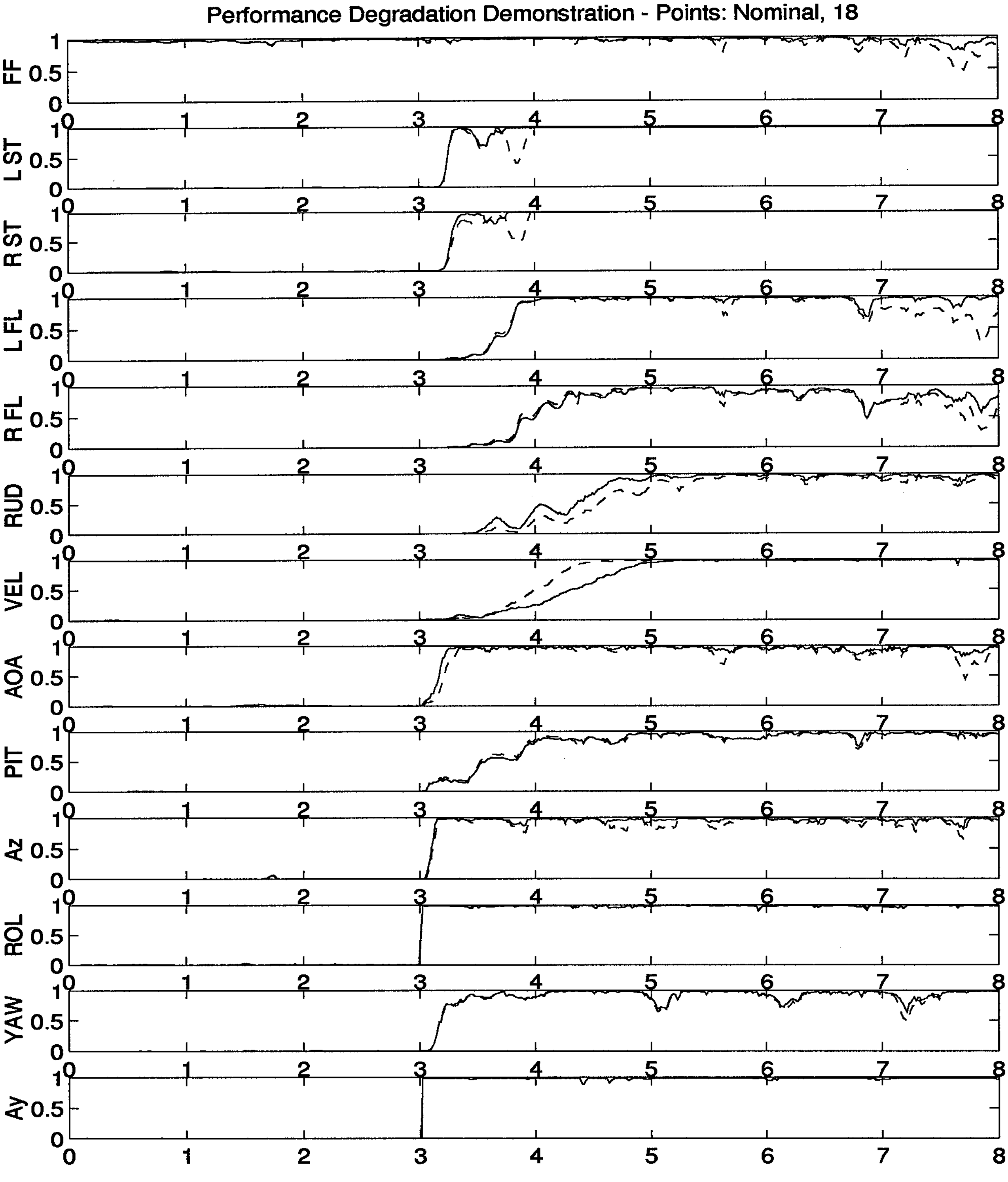


Figure 76. Envelope Radial Line 6: – Nominal, - - Env pt #18

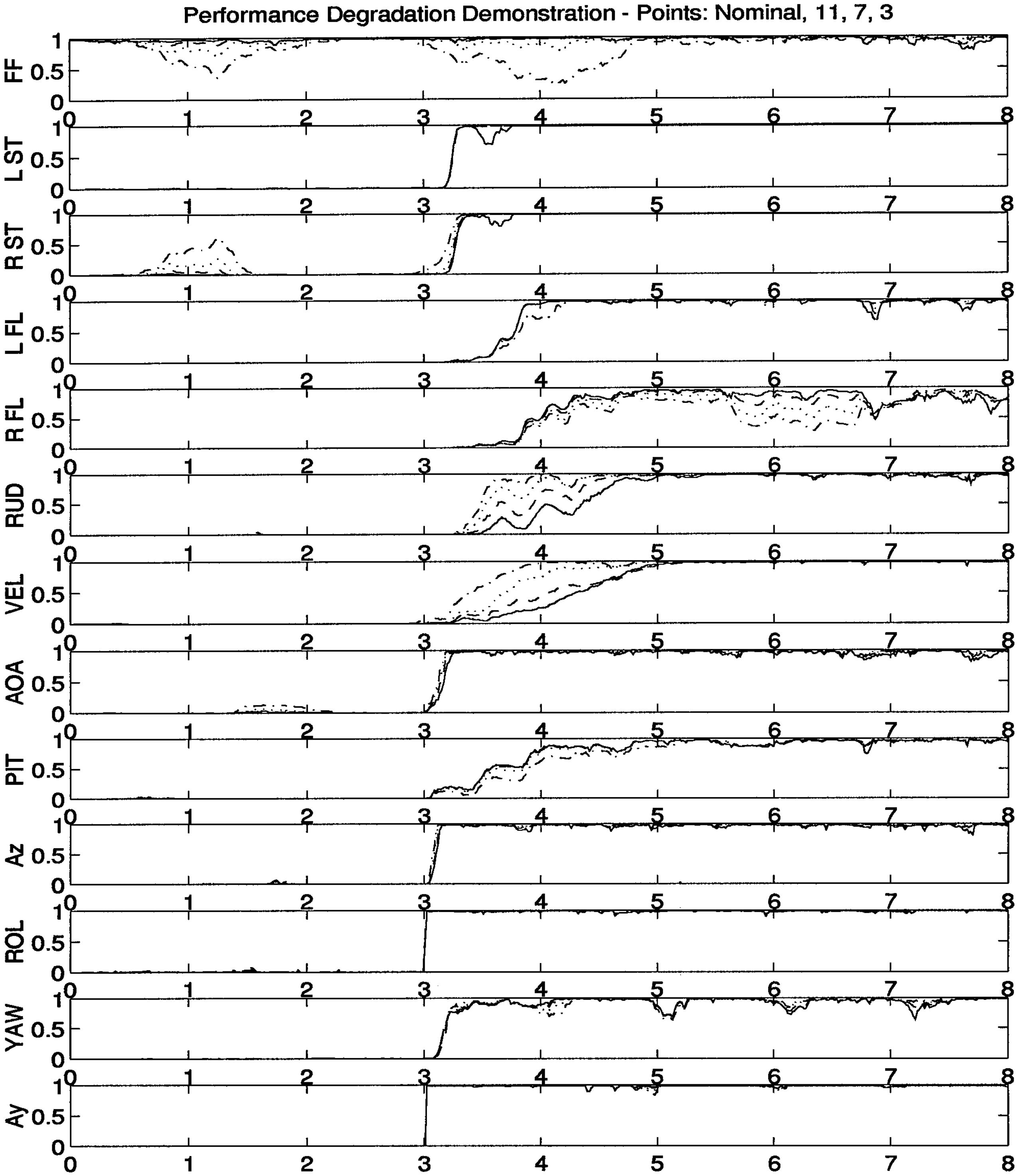


Figure 77. Envelope Radial Line 7: – Nominal, - - Env pt #11, ·· Env pt #7, -·- Env pt #3

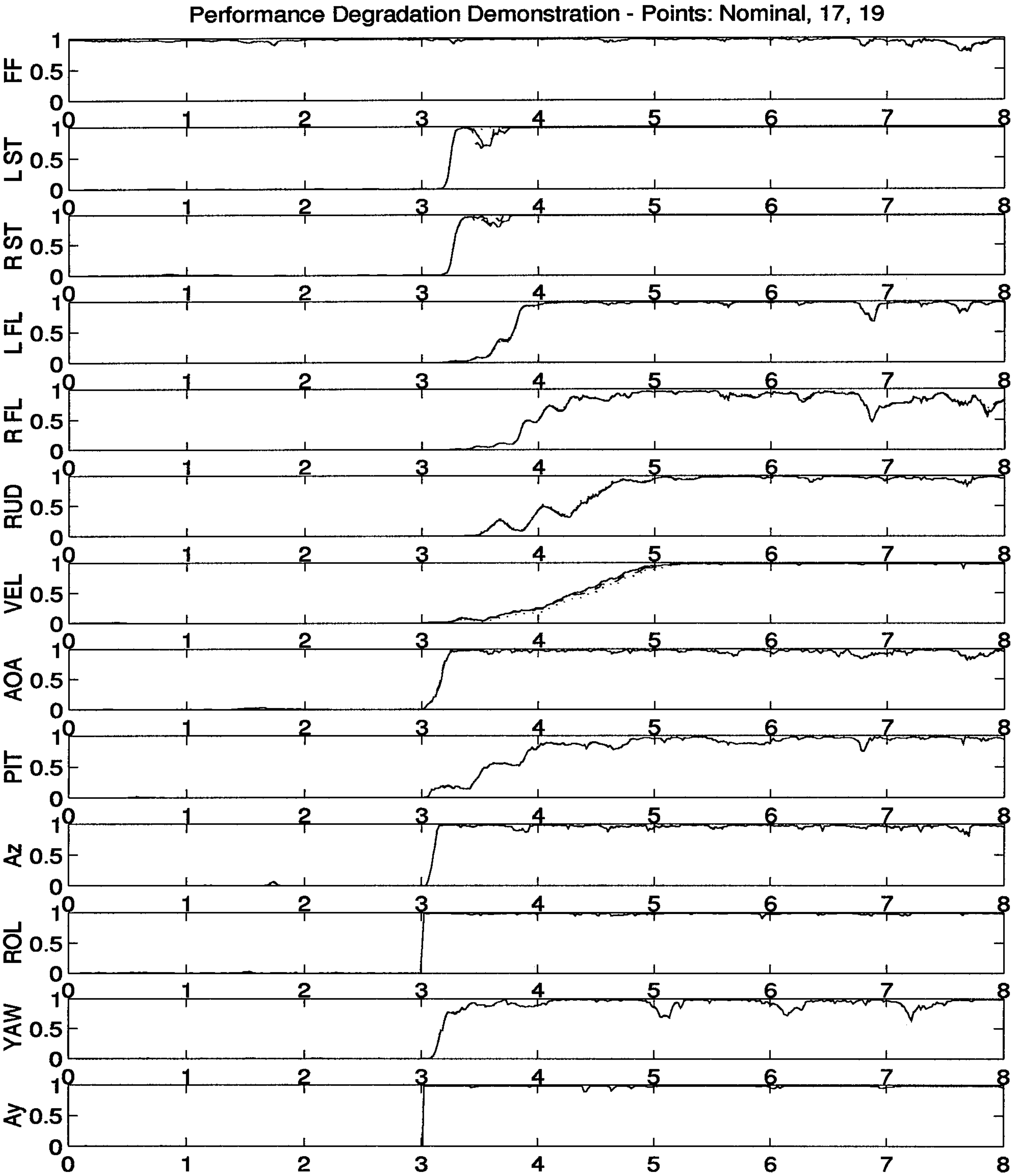


Figure 78. Envelope Radial Line 8: – Nominal, - - Env pt #17, ·· Env pt #19

Figures 75 and 78, on the other hand, do indicate that there is a finite envelope of operation in which the algorithm will continue to exhibit adequate performance. Careful attention to their location within the envelope will reveal that they lie in a line of *approximately* constant dynamic pressure, which supports the good performance maintained. If desired, more points could be selected and evaluated to get the exact shape of this region. Figures 76 and 77 continue to support the false alarm conjecture. Figure 76 has significant false alarms at the end of the run and Figure 77 completely degrades in the beginning and middle of the run; both show that capabilities to detect true failures are maintained. Again, with the exception of our historically difficult failures which show some performance degradation here, the algorithm is continuing to maintain its failure isolation capability. In fact, extremum point 24 in Figure 72 is the only point at which this ability shows any sign of breakdown.

These figures demonstrate the characterizations discussed in detail above. While it appears good that the algorithm's failure identification ability has been retained, the false alarms caused by off-nominal conditions really are the overriding concern. In this application, any false alarms are simply unacceptable. The very small envelope of acceptable operation, again, appears to preclude a full envelope coverage strategy like that pictorially displayed in Figure 3, Chapter 1. This investigation's results point to gain-scheduling for the VISTA F-16 application.

4.5.2 Answers to Research Questions.

- *How quickly does performance degrade off trim?*

Very. Our envelope showed the onset of significant false alarms at the edges of the following envelope:

$$19000 \text{ ft} \leq \text{Altitude} \leq 22000 \text{ ft}$$

$$.38 \text{ Mach} \leq \text{Velocity} \leq .41 \text{ Mach}$$

- *Is the degradation flight-condition-dependent?*

Unknown. Our research never got to a second nominal design point. It would seem that it is flight condition dependent, but not enough so to change the results described in this section.

- *Is a scheduled MMAE feasible based on the number of discrete trim conditions required?*

No. Clearly, small zones of coverage as observed will lead to too many discrete trim conditions required to make this feasible.

- *Can the scheduling mechanism be tied to the FCS gain scheduling being done at present?*

No and yes. Swapping MMAE algorithms in and out as flight conditions change is not feasible. However, we can develop an interpolation gain scheduling scheme for each of the elemental filters and trim conditions within the MMAE which mirrors what is being currently done in the F-16 FCS and tie it together there.

4.6 Chapter Summary

This chapter provided results of the research performed in this thesis. We started by showing how the filters were detuned to provide good single-failure performance. With that filter design in place, we proceeded to examine dual-failure performance and characteristics of the filter residuals. A section was then added to motivate future MMAC research by demonstrating the MMAE-based control's robustness to sensor failures. Finally, we sought full envelope coverage by first observing the algorithm's performance when operated off the nominal design point. The following chapter provides conclusions drawn from the results in this chapter.

5. Conclusions and Recommendations

5.1 Chapter Overview

This chapter draws some conclusions from the results of Chapter 4, offers lessons learned from the entire research effort, and provides some suggestions for future work in the area. Specifically, Section 5.2 consists of a list of conclusions and Section 5.3 contains a couple of suggestions. The chapter concludes with a summary.

5.2 Conclusions

Having completed this research effort, there are a number of items which warrant some concluding remarks. Unfortunately, they cover and overlap all aspects of the preceding work, from trend projection, to lessons learned, to application philosophy, and as such, defy any simple categorization. We therefore, elect to simply present them here in no particular order, relying instead on the accompanying explanation to place a level of relative importance upon them.

5.2.1 MMAE is a Viable Failure Detection Algorithm for Real World Implementation. The results presented in Chapter 4 clearly show that the MMAE algorithm can successfully and reliably detect actuator and sensor failures in a fully nonlinear, higher order environment. Mismatches between our reduced order, linearized, design model and the real world can in large part be accounted for and mitigated through the addition of pseudonoise. Whether an upper ceiling exists where model mismatch cannot be overcome by pseudonoise has not been determined, but is likely. We did find that such detuning costs us speed of probability convergence, sensitivity to certain failures, predictable residual characteristics, and possibly some distinction between left and right channels of the same type of actuator. To the extent that the depth of simulation provided by the SRF VISTA F-16 approaches real world performance (or at least that it approximates the type and level of mismatch which might be expected), our results predict this MMAE algorithm to be a viable failure detection system worthy of implementation on the real world (infinite order)

F-16. Detuning for model mismatch also demonstrates general algorithm flexibility which makes the MMAE attractive for other applications as well.

5.2.2 Algorithm Design Process is Application Oriented. The major recurring theme throughout this work is the need for application-dependent specifications. One cannot, in general, "have his cake and eat it too." In engineering, we often successfully reach some goals only to the detriment of others. In this research alone, we have observed the tradeoff between false alarms and missed alarms brought about by model mismatch and detuning through the addition of pseudonoise. Similarly, there also exists a tenuous balance between false alarms and missed alarms based on the amount of state excitation in the system. We've also seen how improving probability convergence performance degraded failure detection capabilities within the residuals. Clearly, tradeoffs will always exist, but we suggest that where we've already had to accommodate some model mismatch, the margin of error in these tradeoffs become become much smaller. The number of accomplishable goals will be reduced. What is ultimately required in this and other research efforts is a set of well conceived specifications toward which an engineer can design. For the most part, we've demonstrated the kinds and levels of performance which the MMAE is capable and the ability to enhance various aspects of performance via tuning of parameters in the MMAE structure. Specific observable performance may well depend directly on the specific application dependent requirements

5.2.3 Full Envelope Coverage will Require Filter/Trim Gain Scheduling. The prohibitively small area of F-16 flight envelope which our nominal design was able to cover, quickly eliminates the feasibility of using a discrete number of algorithms for full envelope coverage. The critical issue is, of course, model mismatch, which we are inherently increasing by taking the design off nominal. Our method of accommodating model mismatch, addition of pseudonoise, may appear to be a possible way of widening the area of coverage, but recall that we've already done a substantial amount of detuning. In fact, any more detuning and we were running the risk of not detecting a pitch rate failure. It's an obvious conclusion then, that full envelope coverage will require gain scheduling.

Very closely related is the issue of trim variables. As pointed out in Chapter 4, performance of the MMAE depends a great deal on the trim values it uses as nominals. The design model is based on perturbations about a nominal within small angle assumptions. The negative effects of misplacing these nominals will almost instantly result in false alarming. When flight control eventually gets tied to these failure declarations, as in MMAC, incorrect failure declarations can result in instability and loss of aircraft. Any gain scheduling scheme designed for full envelope coverage must also include and account for these varying nominals.

5.2.4 Purposeful Excitation Must Not be Too Violent. Bearing in mind again that the linearized, reduced order, design models are perturbation-based and rely on small angle assumptions, it almost seems obvious to conclude that purposeful excitation mustn't be too violent. This effort though, was the first in this line of research to observe this axiom in practice. We've known that performance is linked positively to the level of system excitation. The more excitation, it appeared, the better the performance. However, now that the MMAE algorithm is immersed in a higher order, fully nonlinear truth model, the magnitudes of such purposeful dither can break down the validity of our design assumptions and add to model mismatch. Although not explicitly reported within our results, the dithering scheme used, which was also employed in past research, was actually quite close to *inducing* false alarms (26). Had we attempted to improve single failure performance by increasing dither, we would have hurt rather than helped our cause. This conclusion is based on observations made only on dither magnitude. Similar observations concerning dither frequency were not made, but it stands to reason that similar logic applies. As frequency is increased, we may for example, experience problems when actuator rate limits begin to impose themselves. Our mismatch will again be exacerbated by unmodeled effects. The overall conclusion to be drawn in this area then, is that when artificially stimulating the system for performance improvement, care must be taken not to increase unmodeled mismatch by violating conditions on which simplifying design assumptions have been made.

5.2.5 Detuning May Alleviate Need for Dither Scheduling. It has been proposed that MMAE dual failure performance can be improved by scheduling dither on the type of first failure declared (26). Again, when the first failure is an actuator, the current dither scheme's loss of ability to excite system state adequately can negatively impact second failure performance. Our results, though, show good dual failure performance possible with detuning instead of scheduling. Perhaps further, careful, detuning with dual failure performance in mind, may eliminate the need to design and employ any dither scheduling at all. Recall that our dual failure results were based on the single-failure scenario filter design which was not optimized by any means because we lacked specifications. In light of the fine effects this detuning had on dual-failure performance, it looks like one ought to be able to utilize detuning as a tool for performance improvement in declaring the second failure as well as the first. Further, if necessary, this could be done in unison with some dither scheduling for further improvement.

5.2.6 Results Reported Represent Worst Case Scenario. As we demonstrated in Chapter 4, Section 4.2.3, failure detection could be affected by position of the perturbation variable within its flight trajectory. This is especially true of the aircraft "slow variables" such as velocity and angle of attack. Recall that in the sample periods immediately following 3.0 seconds, perturbation velocity was at or near zero. This lack of any significant value for the sensors to sense, resulted in slow identification of that failure mode. An extension of the logic concerning this phenomenon allows us to conclude that failure insertion at 3.0 seconds not only represents the worst case scenario for that failure mode, but for all failure modes as well. Consider that the MMAE uses all of its sensors to assist in decision making, even though some effects appear directly in certain sensors more so than in others. Accepting that premise, we again note that at and around 3.0 seconds, the velocity sensor is providing little, and at best ambiguous, information. The algorithm is essentially operating with six sensors instead of seven during that period. We therefore conclude that our results therefore represent the worst case scenario for this particular implementation.

5.2.7 Flight Control System Works Against Effective Dither. Although briefly discussed in Chap 4, Section 4.2.2.1, we wished to reiterate the negative role played by the F-16 FCS in our overall failure detection performance and point out that implementation difficulties can come from sources other than the models themselves. The aileron to rudder interconnect (ARI) made it impossible for us to work the rudder and flaperons continuously, simultaneously, and separately. Since the F-16 is not a boat, the rudder's only real purpose is in turn coordination. It's deflections are therefore inextricably tied to flaperon deflections. In our case, no separate rudder dither was required because dither in the flaperons cause plenty of rudder dither. The ARI along with requirements of subliminality led to the mediocre across-the-board flaperon performance we observed. Another FCS limitation which might be a problem, but we did not experience, is the sharing of primary control authority between stabilators and flaperons. The FCS assumes the pilot is asking for a rate not a deflection; therefore if a rate is requested which calls for more deflection in one set of surfaces than is available, it obtains assistance from redundant surfaces. The point is, that a dither scheme for the F-16 may not give the designer what he anticipates due to these types of nonlinearities. More generally, we can conclude from our experiences with the FCS that other system limitations may exist from application to application which will impact the MMAE algorithm's ability to operate effectively.

5.2.8 Continuous Dither Excitation May be Unnecessary. A rather strong philosophical argument can be made against the use of a continuous dither signal. While we've generated no quantitative data to support the argument, our experiences have led us to conclude in its favor. Consider that the purpose of dithering is to excite the state variables artificially for the purpose of detecting and isolating failures during benign, steady level, flight. Menke demonstrated that the MMAE detects and isolates failures quite well when system excitation comes directly from pilot command (26). Low dynamic pressure, steady level, benign flight was determined to be the difficult scenario for the MMAE. In fact, the research projects which followed Menke invested a majority of time ensuring identification capabilities during steady level flight. This involved determining a single, optimal, and subliminal, dither scheme and as we've shown, considerable tuning. This is quite a painstaking process, especially when there are questions concerning its necessity.

There are no obvious requirements for continuous dither. Benign flight is simply that, *benign.* Why then unnerve the pilot with "subliminal" jostling, or reduce the life expectancy of aircraft actuators just to get continuous coverage during benign flight? Even assuming a failure did occur sometime during benign flight, it would almost immediately show itself upon pilot issued commands. Surely we recognize that finding out about an actuator failure after engaging the enemy is probably too late, but that doesn't argue for *continuous* dither. Rather, it argues for a way to check out system integrity prior to engagement and such a capability can exist in a Built-in-Test (BIT). A BIT capability would allow the pilot to initiate a MMAE checkout whenever he desires and more importantly, before entering combat. We point out that this checkout wouldn't have to suffer the restrictions of optimality and subliminality which currently bind the design. The test could be a short duration, sequential working of the three actuator groups, featuring good isolation of left and right channel effects: a better test! This research has shown that it is possible to monitor for failures continuously during benign flight, but in light of sound philosophical reasoning against it, we conclude that continuous dither is in fact, unnecessary.

5.2.9 Robustness to Sensor Failure a Capability Worth Advancing. One very nice byproduct of this research was the demonstration of sensor failure robustness. It points very optimistically in the direction of MMAC which this MMAE research precurses. The conclusion we would like to draw is that, if this technology were to be implemented into the F-16 in phases, the algorithm would already have virtue and utility simply as a sensor measurement estimator. Of course, bigger and better utility is in store for the algorithm which would follow in the phased implementation. This feature, however, deserves present attention and we conclude, stands on its own merits.

5.3 Recommendations

There are many directions one could take to further this line of research. We believe, however, that the direction which ought to be taken is directly towards implementation. Enough time has been spent in the past proving the concept. This research effort has taken a significant step from academia towards real world implementation. Our MMAE algorithm proved quite capable when put up against the most sophisticated F-16 simulation in the U.S. Air Force. The next logical step, then, is to work towards actual implementation. The following sections describe work which needs to be accomplished en route.

5.3.1 Design and Implement Gain Scheduling System. Now that we've come to the conclusion that gain scheduling is required for full envelope coverage, the next logical step is to design the gain scheduling system and software. Typically, this involves designing discrete MMAE algorithms at various points across the envelope, then curve fitting their gains (including trims). The curves which result would then provide a continuous relationship between algorithm gains and aircraft location within the flight envelope via interpolation. Exactly how many discrete designs would be necessary is as yet unknown since we've only designed for one point in this effort. A conservative approach though, would start with a manageably coarse discretization and add points as required for good performance. Accomplishing this task would be beneficial for possible demonstration on flight simulators as well as eliminating the largest impediment to real world implementation.

5.3.2 Develop the BIT Scheme. As discussed above, continuous sinusoidal dither may not be an operationally good idea. In fact, we feel it's not unlikely that some decision makers may have resistance to the idea altogether. With such foresight, we recommend that a BIT scheme be designed and implemented in the near future. The BIT should be a minimum duration, sequential, shake-up of the stabilators, flaperons, and rudder. Since the test is not continuous, each portion of the shake-up can isolate affected individual channels. Sensor failures need not have tests specifically designed for their identification because their failure effects will show up at any time during the actuator isolation process. Accomplishing this task now will give us the benefit of being able to prove the theory and work out any "bugs" early, as well as provide more options to the decision makers.

5.3.3 Design the MMAC. Concurrent with the aforementioned recommendations, work can begin on designing the elemental controllers displayed in Chapter 1, Figure 6. At this point, the controllers can be designed such that stability and performance are maintained at our nominal point when single and dual failures are inserted. Work accomplished in this direction can then be married easily with the results of our other recommendations and be dovetailed into a finished product in the future. The ultimate goal of this line of research is still a Multiple Model Adaptive Controller.

5.3.4 Others. Other items which also warrant attention are, interfacing a full envelope configuration to the graphics package *Aviator*, and implementing the algorithm on a parallel processing computer. Both of these recommendations will be beneficial in that they facilitate further demonstration capabilities within the computer simulation environment. Like the recommendations elaborated upon above, these offer positive steps forward, designed to promote acceptance by decision makers. Pursuit of real world implementation, however, encourages that greater priority be placed on the first two.

5.4 Chapter Summary

This chapter has drawn conclusions based on experiences and results of work performed throughout this thesis research process. The MMAE algorithm can provide single and dual actuator and sensor failure identification capabilities to the F-16 and also robustify the FCS to sensor failures.

The positive results beg continued efforts in this area with the ultimate goal being MMAC. For now, though, an implementable version of the MMAE for use on the F-16 ought to be the top priority. Appendices A through E contain good information which supplements the results and discussion provided in this text.

The $\mathbf{\Phi}$, $\mathbf{B}_d$, and $\mathbf{Q}_d$ matrices can be generated via the equations shown below:

$$\mathbf{\Phi} = e^{\mathbf{A}\Delta t}$$

since $\mathbf{A}$ is constant,

$$\mathbf{B}_d = \int_{t_{i-1}}^{t_i} e^{\mathbf{A}(t_i-\tau)}\mathbf{B}(\tau)d\tau$$

and

$$\mathbf{Q}_d = \int_{t_{i-1}}^{t_i} e^{\mathbf{A}(t_i-\tau)}\mathbf{G}\mathbf{Q}\mathbf{G}^T e^{\mathbf{A}(t_i-\tau)T} d\tau$$

The $\mathbf{\Phi}$ and $\mathbf{B}_d$ matrices were generated using the Matlab function *c2d*, and $\mathbf{Q}_d$ was intercepted in the middle of Matlab function *lqed*. The values used in this thesis are then:

$$\mathbf{\Phi} = \left[\begin{array}{ccccc}
1.0000 & -1.7742 \times 10^{-7} & 1.4261 \times 10^{-4} & 1.5561 \times 10^{-2} & \cdots \\
-4.9582 \times 10^{-1} & 9.9993 \times 10^{-1} & 5.0685 \times 10^{-1} & -1.0901 & \cdots \\
-2.0702 \times 10^{-4} & -3.4439 \times 10^{-6} & 9.9390 \times 10^{-1} & 1.5466 \times 10^{-2} & \cdots \\
1.3841 \times 10^{-5} & -2.2689 \times 10^{-5} & 1.8208 \times 10^{-2} & 9.9190 \times 10^{-1} & \cdots \\
0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & \cdots
\end{array} \right.$$

$$
\begin{array}{cccccc}
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 1.0000 & -2.1324 \times 10^{-3} & 1.5430 \times 10^{-2} & 2.7091 \times 10^{-3} & \cdots \\
\cdots & 1.2057 \times 10^{-3} & 9.9755 \times 10^{-1} & 2.6585 \times 10^{-3} & -1.5535 \times 10^{-2} & \cdots \\
\cdots & -1.6947 \times 10^{-4} & -2.7953 \times 10^{-1} & 9.7512 \times 10^{-1} & 2.6798 \times 10^{-3} & \cdots \\
\cdots & 2.7431 \times 10^{-5} & 4.5430 \times 10^{-2} & -5.9518 \times 10^{-4} & 9.9525 \times 10^{-1} & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots
\end{array}
$$

$$
\left.
\begin{array}{cccccc}
\cdots & -2.0871 \times 10^{-4} & -2.0871 \times 10^{-4} & 1.0212 \times 10^{-5} & 1.0212 \times 10^{-5} & 0 \\
\cdots & 2.9463 \times 10^{-2} & 2.9463 \times 10^{-2} & -3.0344 \times 10^{-3} & -3.0344 \times 10^{-3} & 0 \\
\cdots & -6.8725 \times 10^{-4} & -6.8725 \times 10^{-4} & -1.3318 \times 10^{-4} & -1.3318 \times 10^{-4} & 0 \\
\cdots & -2.5740 \times 10^{-2} & -2.5740 \times 10^{-2} & 1.2589 \times 10^{-3} & 1.2589 \times 10^{-3} & 0 \\
\cdots & 5.2525 \times 10^{-4} & -5.2525 \times 10^{-4} & 7.0723 \times 10^{-4} & -7.0723 \times 10^{-4} & 3.0174 \times 10^{-4} \\
\cdots & -6.8559 \times 10^{-5} & 6.8559 \times 10^{-5} & 1.0942 \times 10^{-4} & -1.0942 \times 10^{-4} & 4.2722 \times 10^{-4} \\
\cdots & 6.3307 \times 10^{-2} & -6.3307 \times 10^{-2} & 8.6790 \times 10^{-2} & -8.6790 \times 10^{-2} & 3.9919 \times 10^{-2} \\
\cdots & 7.4221 \times 10^{-1} & -7.4221 \times 10^{-3} & 9.3274 \times 10^{-4} & -9.3274 \times 10^{-4} & -1.6505 \times 10^{-2} \\
\cdots & 8.0352 \times 10^{-1} & 0 & 0 & 0 & 0 \\
\cdots & 0 & 8.0352 \times 10^{-1} & 0 & 0 & 0 \\
\cdots & 0 & 0 & 8.0352 \times 10^{-1} & 0 & 0 \\
\cdots & 0 & 0 & 0 & 8.0352 \times 10^{-1} & 0 \\
\cdots & 0 & 0 & 0 & 0 & 8.0352 \times 10^{-1}
\end{array}
\right]
$$

$$\mathbf{B}_d = \begin{bmatrix} -1.5502 \times 10^{-5} & -1.5502 \times 10^{-5} & 7.5865 \times 10^{-7} & 7.5865 \times 10^{-7} & 0 \\ 2.7735 \times 10^{-3} & 2.7735 \times 10^{-3} & -3.1453 \times 10^{-4} & -3.1453 \times 10^{-4} & 0 \\ -6.9859 \times 10^{-5} & -6.9859 \times 10^{-5} & -1.5512 \times 10^{-5} & -1.5512 \times 10^{-5} & 0 \\ -2.9220 \times 10^{-3} & -2.9220 \times 10^{-3} & 1.4296 \times 10^{-4} & 1.4296 \times 10^{-4} & 0 \\ 3.9069 \times 10^{-5} & -3.9069 \times 10^{-5} & 5.2609 \times 10^{-5} & -5.2609 \times 10^{-5} & 2.2454 \times 10^{-5} \\ -8.8819 \times 10^{-6} & 8.8819 \times 10^{-6} & 7.9676 \times 10^{-6} & -7.9676 \times 10^{-6} & 4.1049 \times 10^{-5} \\ 7.2079 \times 10^{-3} & -7.2079 \times 10^{-3} & 9.8828 \times 10^{-3} & -9.8828 \times 10^{-3} & 4.5479 \times 10^{-3} \\ 8.4295 \times 10^{-3} & -8.4295 \times 10^{-3} & 1.0689 \times 10^{-4} & -1.0689 \times 10^{-4} & -1.8723 \times 10^{-3} \\ 1.9648 \times 10^{-1} & 0 & 0 & 0 & 0 \\ 0 & 1.9648 \times 10^{-1} & 0 & 0 & 0 \\ 0 & 0 & 1.9648 \times 10^{-1} & 0 & 0 \\ 0 & 0 & 0 & 1.9648 \times 10^{-1} & 0 \\ 0 & 0 & 0 & 0 & 1.9648 \times 10^{-1} \end{bmatrix}$$

$$\mathbf{Q}_d = \left[\begin{matrix} 5.8216 \times 10^{-12} & 2.0303 \times 10^{-8} & -2.5638 \times 10^{-10} & 5.5642 \times 10^{-10} & \dots \\ 2.0303 \times 10^{-8} & 1.6184 \times 10^{-4} & -2.1959 \times 10^{-6} & 2.6039 \times 10^{-6} & \dots \\ -2.5638 \times 10^{-10} & -2.1959 \times 10^{-6} & 3.0016 \times 10^{-8} & -3.2891 \times 10^{-8} & \dots \\ 5.5642 \times 10^{-10} & 2.6039 \times 10^{-6} & -3.2891 \times 10^{-8} & 7.1224 \times 10^{-8} & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \end{matrix} \right.$$

$$\begin{array}{cccccc}
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 1.1809 \times 10^{-9} & 8.9788 \times 10^{-10} & 1.1579 \times 10^{-7} & -1.8551 \times 10^{-8} & \cdots \\
\cdots & 8.9788 \times 10^{-10} & 4.0991 \times 10^{-8} & 1.0159 \times 10^{-7} & -1.0964 \times 10^{-8} & \cdots \\
\cdots & 1.1579 \times 10^{-7} & 1.0159 \times 10^{-7} & 1.5242 \times 10^{-5} & -2.4339 \times 10^{-6} & \cdots \\
\cdots & -1.8551 \times 10^{-8} & -1.0964 \times 10^{-8} & -2.4339 \times 10^{-6} & 3.9969 \times 10^{-7} & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & \cdots
\end{array}$$

$$\left.\begin{array}{cccccc}
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0 \\
\cdots & 0 & 0 & 0 & 0 & 0
\end{array}\right]$$

Appendix B. VISTA SRF Parameter Values

This appendix is provided to facilitate research repeatability. The following table displays the parameter value settings which were used during MMAE research. These values are set while in the interactive mode of SRF. The interactive mode is run by typing *xsrf* at the prompt within the directory containing the SRF simulation. Typing *vista* at the first menu, then choosing Option 1 will avail these parameters for setting prior to runtime. The nomenclature used in Table 11 entries mirrors the menus displayed in the SRF interactive mode. (Entries such as "parm" and "APRET" appear exactly as they do in the menus). Important information given here is contained in the "value" column. Consult the SRF User's Manual for a complete description of the parameters as well as all possible settings (12).

Table 11. SRF Parameter Values

parm	description	value
outfile	Name of APRET output file	"vista.output"
ofiletyp	output file format	"apret"
eqmot	Equation of motion to be used	"gd"
stoptime	Simulation duration (sec)	8.0
saverate	Data recording sample rate (Hz)	32.0
afmrate	Airframe model iteration rate	128.0
DFCSrate	DFCS iteration rate	64.0
trimic	Trim with initial conditions	"no"
ctlfile	Name for SEL CTL file	(null value)
altitude	Initial altitude (ft)	20000
Mach	Initial airspeed	0.4
Xpos	Initial aircraft X axis (ft)	0.0
Ypos	Initial aircraft Y axis (ft)	0.0
heading	Initial A/C true heading (deg)	0.0
gamma	Initial glide slope angle (deg)	0.0
thrcntrl	Throttle control type	"constant"
stores	Stores configuration	"standard"
fltcond	Flight Condition	"up+away"
atmos	Type of day for atmosphere model	"standard"
cmdfile	Name of command file	(null value)
commands	Pilot commands	(null value)
cmdmagn	Command magnitude	(null value)
cmdtime	Start time for each command	(null value)
cmddur	Duration of each command	(null value)
gear	Initial landing gear setting	(null value)
geartime	Time to toggle gear setting	(null value)
spdbrake	Initial speed brake setting	(null value)
spdtime	Time to toggle speed brake	(null value)
flap	Initial flap setting	(null value)
flaptime	Time to toggle flap setting	(null value)
mxfile	Name of MATRIXxfile for linear model generation	"trims.dat"
multwgt	A/C weight multiplication factor for linear model generation	1.0
lse	Local slope extraction	"no"
saveit	Debug data save flag	"no"
savefn	Debug data save filename	"debug_iofn.dat"
restorit	Debug data restore flag	"no"
restorfn	Debug data restore filename	"debug_iofn.dat"

Appendix C. SRF Modification Table

The following page contains a table providing a listing of SRF FORTRAN files modified, created, or added for use in this thesis research. A brief description of the changes are also given.

Table 12. Modified/Created SRF Files

FORTRAN File	Description of Change
afm.F	- Added alternative calculation of A_n and A_y using equations $-\frac{\bar{U}}{g}(\dot{\alpha} - q) - 1$ and $\frac{\bar{U}}{g}(\dot{\beta} + r)$ - Added sensor failure insertion logic code for A_n and A_y - Added code (unused) which adds full Dryden noise
atr40.F	- Added actuator failure logic code
blend.F	*NEW*: mmae routine which blends the elemental outputs to form the state estimate
buildmx.F	- Added forward velocity variable STATELAT(P_VEL) - Added MatrixX command code to generate **G**, **C**, **D** and our modified **B**
creatvec.F	*NEW*: assembles **u** and $\mathbf{z}_i$ vectors for MMAE, adds sensor noise
c_mixout.inc	- Added shift variables for use in m02.F below
datapool.inc	- Added PGUST, QGUST, RGUST (unused) variables for full Dryden model
failchk.F	*NEW*: mmae routine which calculates elemental probabilites and checks for failure
getcom.F	- Added dither signal code
inimmae.F	*NEW*: mmae routine which reads in the elemental filter data
m02.F	- Added code to delay/offset left and right channel commands for heightened left/right flaperon failure identification
matadd.F	*NEW*: mmae utility routine which adds matrices
matc32.F & matc42.F	*NEW*: mmae utility routines which insert and remove data from three and four dimensional arrays
matcopy.F	*NEW*: mmae utility routine which copies matrices
matml.F	*NEW*: mmae utility routine which multiplies matrices
matsub.F	*NEW*: mmae utility routine which subtracts matrices
mattp	*NEW*: mmae utility routine which transposes matrices
mmae.F	*NEW*: top level mmae program which is called by vista.F
mmaeflags.dat	*NEW*: read by readmmae.F, contains failure insertion data
mmaeinput.inc	*NEW*: .inc file for SRF containing variable declarations and parameter values for mmae/related variables
MMAEVAR.TXT	*NEW*: mmae .txt file containing variable declarations and parameter values
mmvert.F	*NEW*: mmae routine which subtracts nominals from **u** and **z** converts units as necessary
p02.F	- Added software breakout code included in dwgs but not in SRF
p03.F	- Replaces q and A_n variables w/ MMAE generated estimates
p04.F	- Replaces aoa variable w/ MMAE generated estimate
propgate.F	*NEW*: mmae routine to propogate elemental filter estimates
r02.F	- Added software breakout code included in dwgs but not in SRF
r04.F	- Replaces p variable w/ MMAE generated estimate
readmmae.F	*NEW*: reads in failure status data at runtime
restart.F	*NEW*: mmae routine which initializes the algorithm
sensors.F	- Added failure insertion code for all but accel sensor failure
simulate.F	- Added hooks for HOTAS inputs (unused)
sort.F	*NEW*: mmae routine which sorts the probabilites after lower probability bounding has occurred
turb2.F	- Added calculations for Dryden rotational gusts P,Q,R (unused)
update.F	*NEW*: mmae routine to update the elemental filters with measurement information
vista.F	- Added initialization and run calls to mmae.F and creatvec.F - Added logic to call MMAE every other simulation iteration - Added lots of i/o file headers and trailers for data collection
wind.inc	- Added Zero-order Dryden model states - Added Dryden Rotational states (unused)
wind.F	- Inserted my Zero-order Dryden model here
wind_setup.F	- Added Zero-order Dryden model states - Added Dryden Rotational states (unused)
y03.F	- Added software breakout code included in dwgs but not in SRF
y04.F	- Replaces r variable w/ MMAE generated estimate
y05.F	- Replaces A_y variable w/ MMAE generated estimate

Appendix D. Filter Generation Process - Step-by-Step

To facilitate repeatability of this research, the following step-by-step instructions for generating the filter files is provided. Before proceeding, the following software tools and files are necessary.

- Matlab
- MatrixX
- Modified SRF Tool
- filtergen.m
- convert.for

At this point a basic knowledge of Matlab, MatrixX, FORTRAN and SRF is assumed. The reader is referred to references for further information (11–13, 19, 34). Also, Appendix B contains the actual variable values used in the SRF simulation.

Step 1. Create **A**, **B**, **G**, and **H** matrices. The SRF's linearization software module buildmx.f has been modified to generate these matrices when performing its linearization. To get them into a usable form, do the following.

Step 1a. Under Option (1) of the SRF VISTA Menu, set parameter **mxfile** to a filename of your choice: *mxfile = filename.* When you run the simulation, a MatrixX file **filename.dat** will be generated containing the linearized trim data. Among other things, it has the dimensional derivatives in the body axis and MatrixX commands to generate the desired matrices. The commands are based on Equations (30), (32), and (46).

Step 1b. Move **filename.dat** into a MatrixX directory and start up MatrixX. At the prompt, type *exec('filename.dat').* This will automatically create the file **srfdat_mxx.dat** which is in MatrixX backup format and contains the desired matrices. This data must be converted into ASCII format, readable by Matlab.

Step 1c. Compile and execute the fortran program **convert.for**. This can be done on a Unix machine by typing *f77 -o executablename convert.for*, then executing the program by typing *executablename* (whatever name you've given it). The result is an ASCII file called **srfdat.m** which contains the desired matrices and is readable by Matlab.

Step 2. Create **R** and **Q**. No special software is used to generate these two matrices. Rather, they are usually determined empirically a priori and not directly related to any dimensional derivatives. This step requires only that they be of correct dimension and reside in an ASCII file with the name **noises.m**. See the previously generated **srfdat.m** for an idea of what the format should be.

Step 3. Generate the filter files. With **srfdat.m**, **noises.m**, and **filtergen.m** in a Matlab working directory, start up Matlab. At the Matlab prompt, type *filtergen.* This m-file will generate ASCII filter files if the internal variable *bin* is set to zero, or Matlab binary filter files if *bin* is set to one. The switch must be set within **filtergen.m**. NOTE: If simply recreating the results of this thesis, no further work is necessary. 13 banks of filters based on five actuator failures, seven sensor failures, and the no-fail condition are generated. If attempting a different application, **filtergen** and **convert.for** will need appropriate modifications. See internal code documentation for further guidance.

Appendix E. Dual Failure Summary Plots

The following figures provide a condensed form of the 132 dual failure runs whose qualitative results were reported in Chapter 4, Table 8. To avoid reprinting 132 separate figures, the following compromise was reached. Each figure plots all of the combinations of possible secondary failure for a given first failure. The individual plots within each figure are the appropriate channel in which we expect to see secondary failure probability convergence. For example, Figure 79 shows all combinations of left stabilator actuator failure with other secondary failures. The first plot within it is right stabilator actuator channel for the Left Stabilator/Right Stabilator failure combination. This format allows us to see at a glance, how the secondary failures were or were not picked up in the correct channel. Unfortunately, when the algorithm struggles, we cannot see into which channel the probabilities are incorrectly distributed. This will only be a slight drawback, though, because the majority of secondary failures converge quite nicely. These plots are intended to support the qualitative results reported in Table 8.

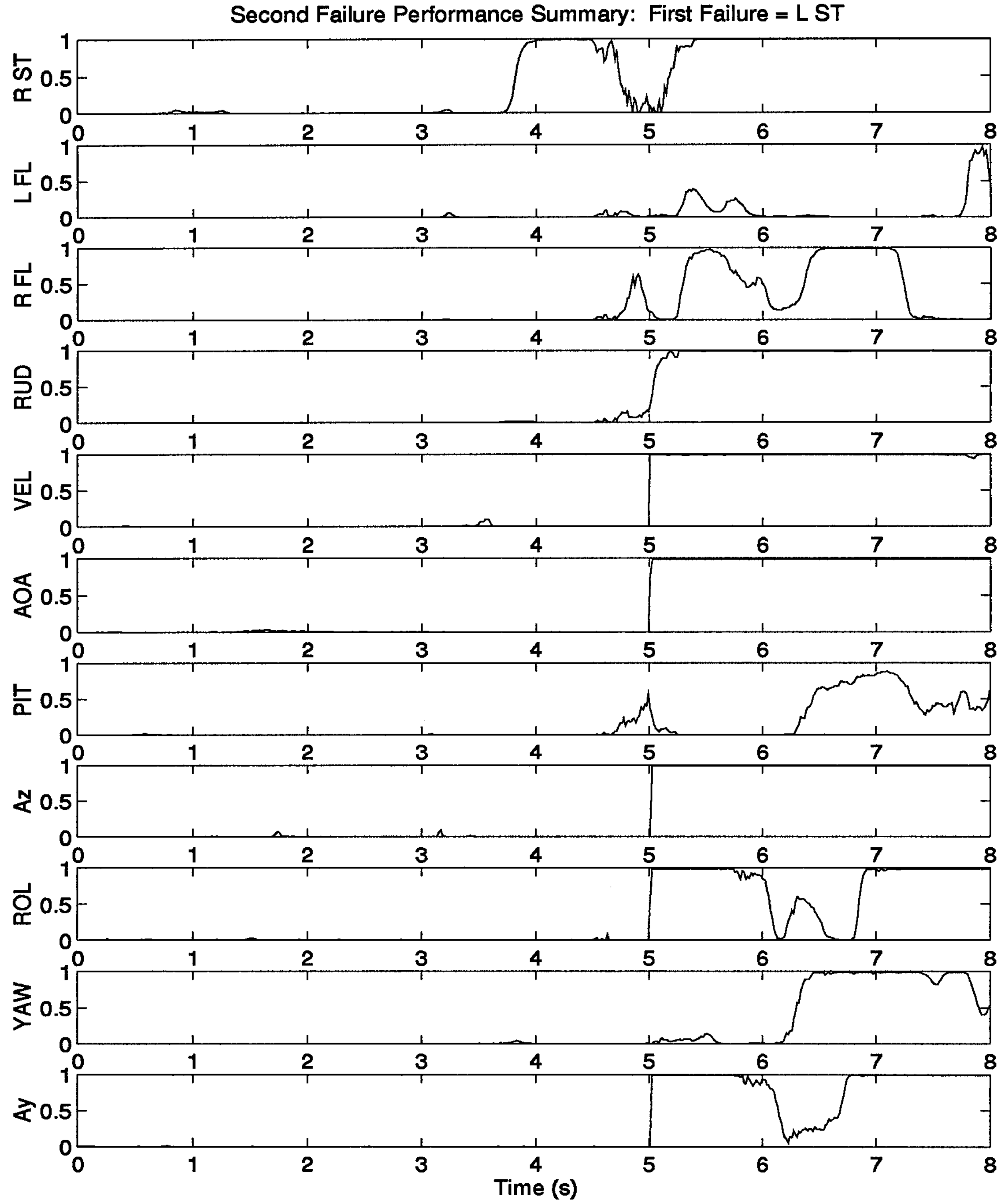


Figure 79. Left Stabilator Actuator Summay Plot - Secondary Failure Induced at 5.0 seconds

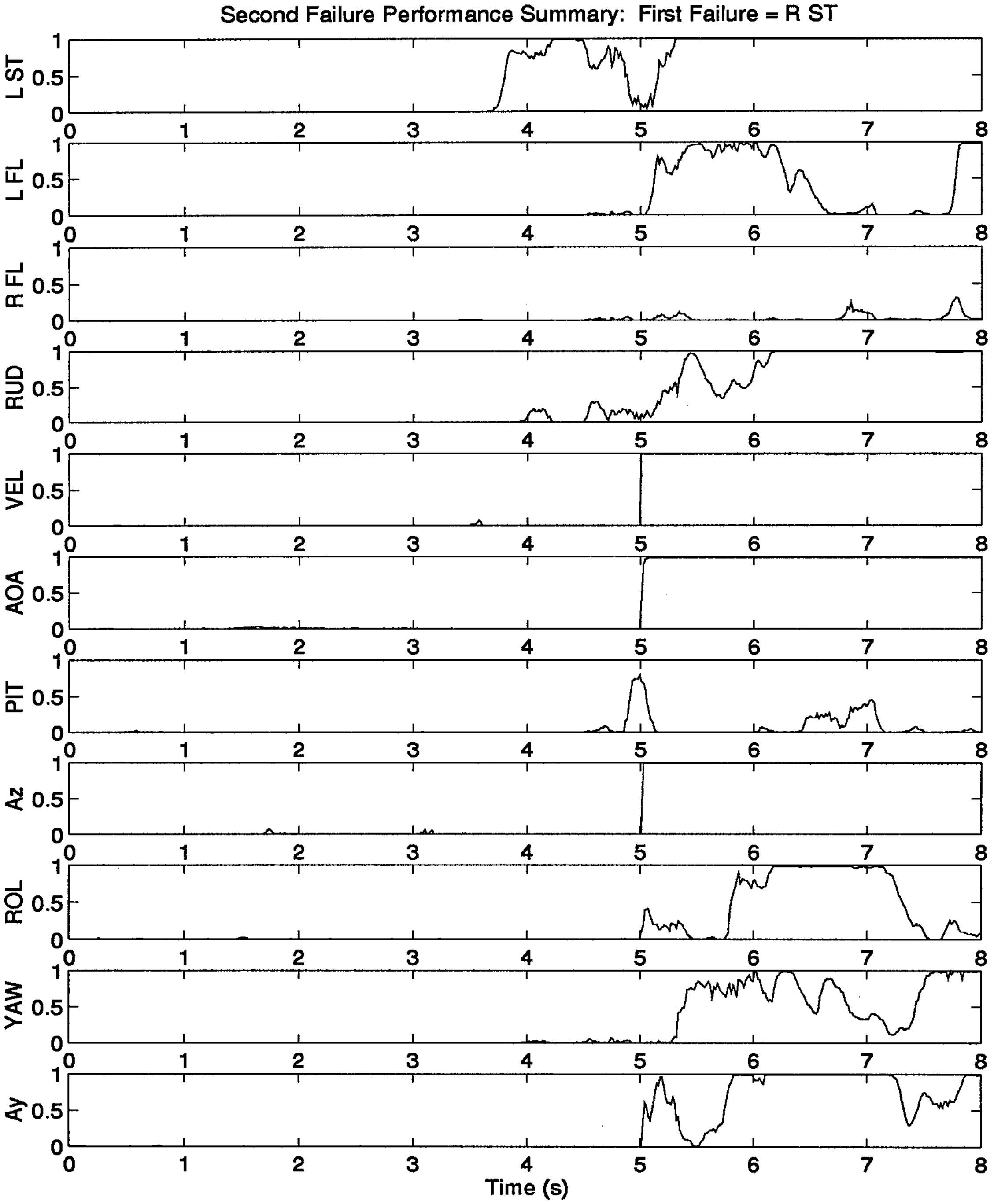


Figure 80. Right Stabilator Actuator Summay Plot - Secondary Failure Induced at 5.0 seconds

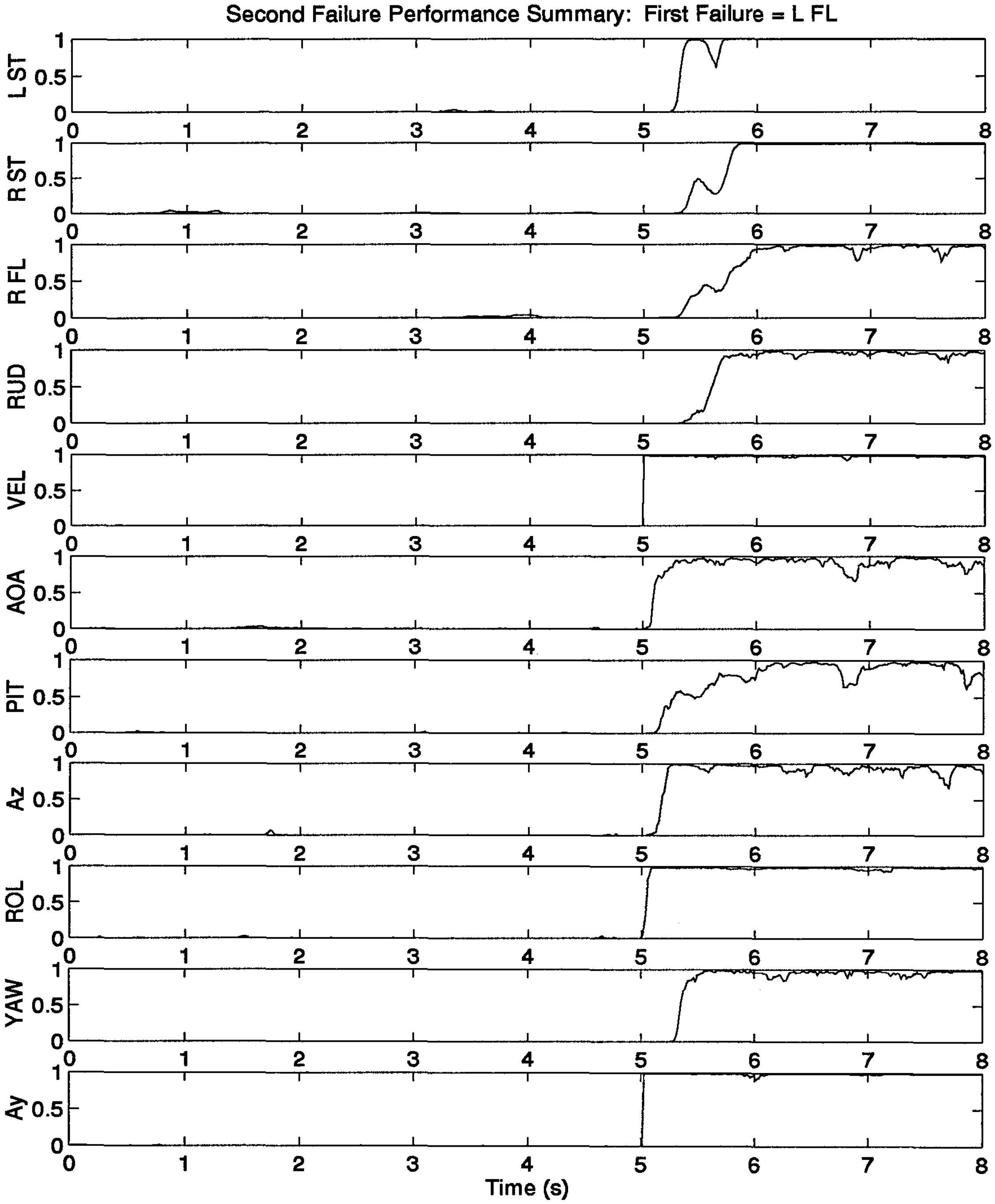


Figure 81. Left Flaperon Actuator Summay Plot - Secondary Failure Induced at 5.0 seconds

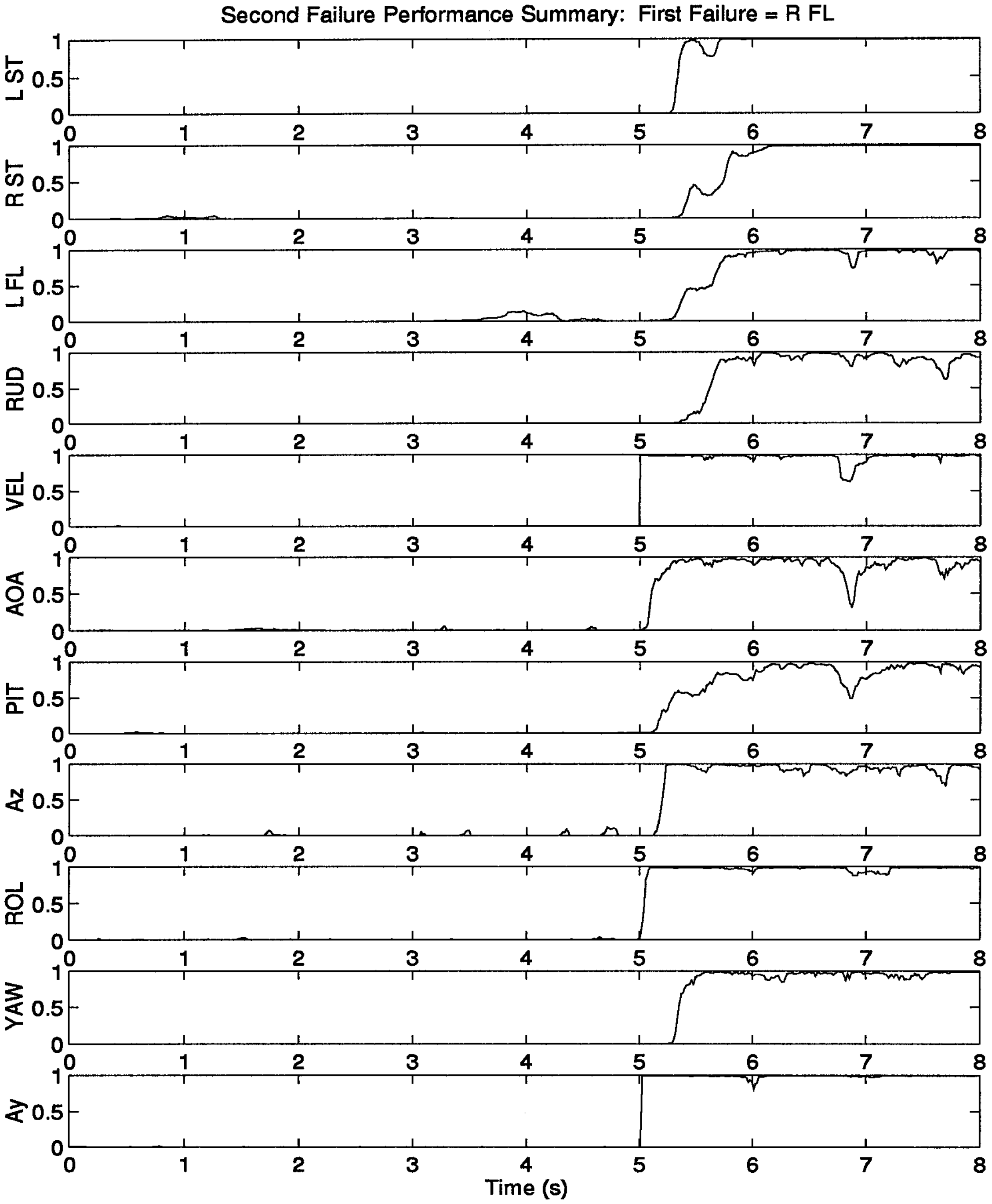


Figure 82. Right Flaperon Actuator Summay Plot - Secondary Failure Induced at 5.0 seconds

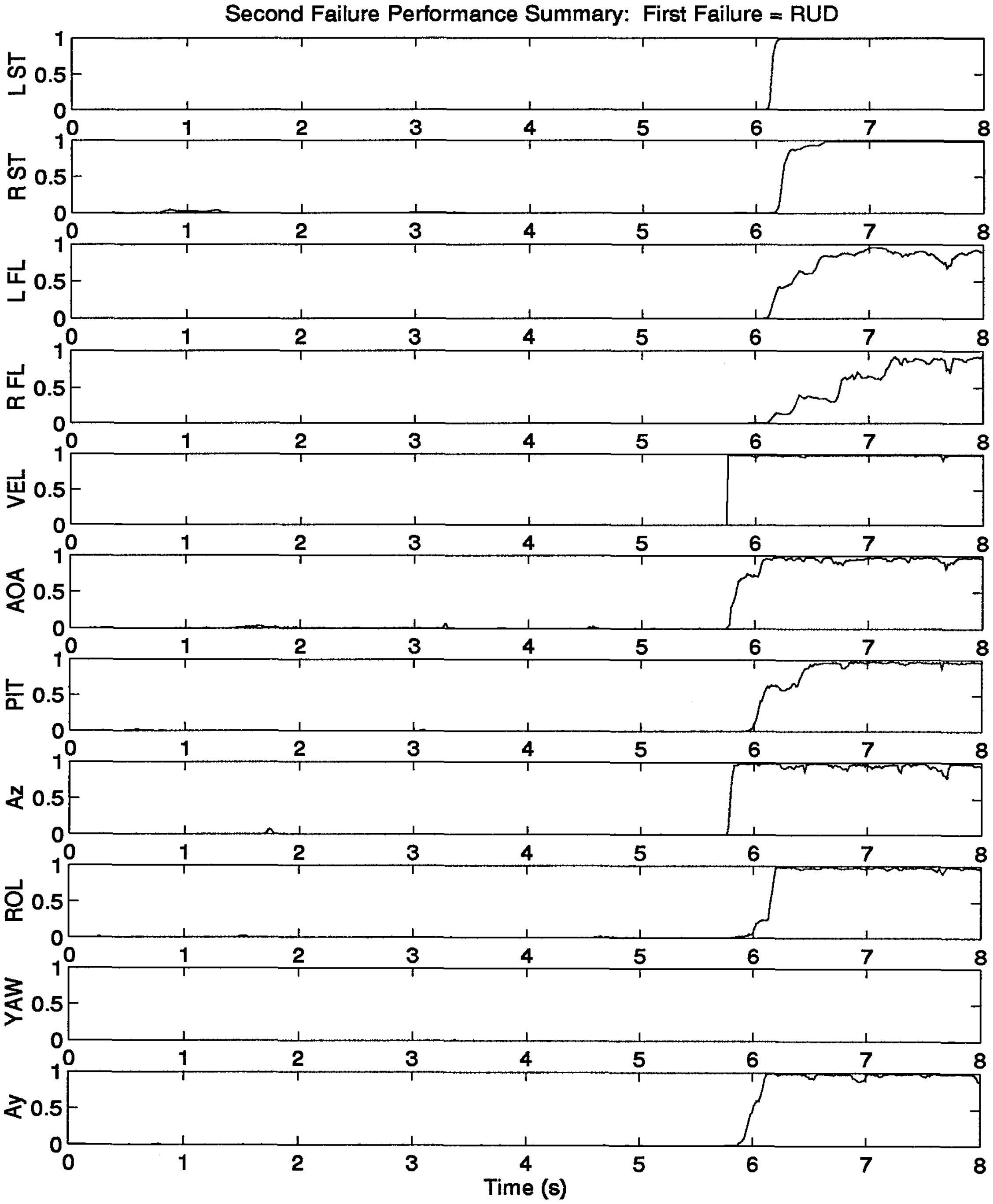


Figure 83. Rudder Actuator Summay Plot - Secondary Failure Induced at 5.7 seconds

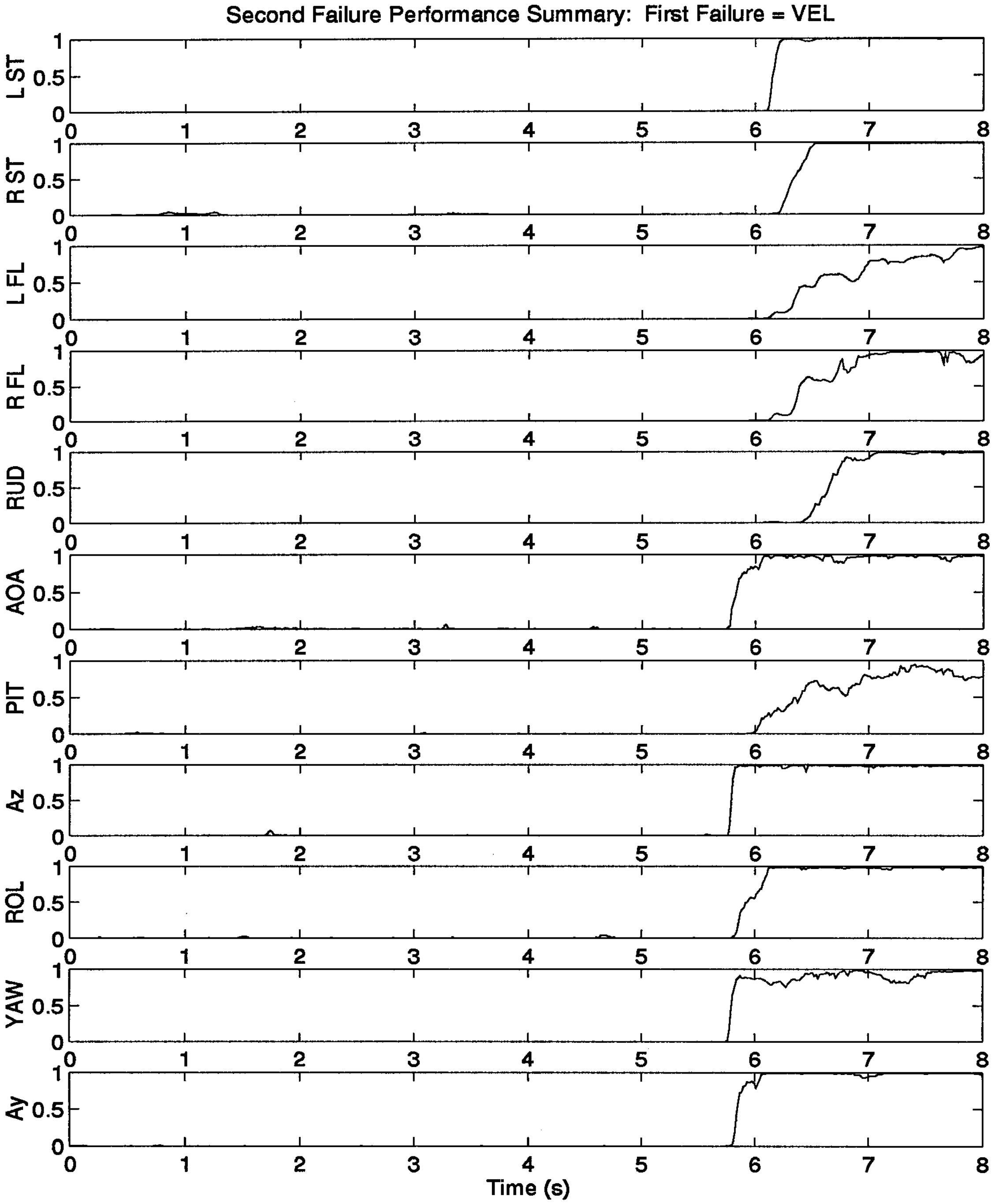


Figure 84. Forward Velocity Sensor Summay Plot - Secondary Failure Induced at 5.7 seconds

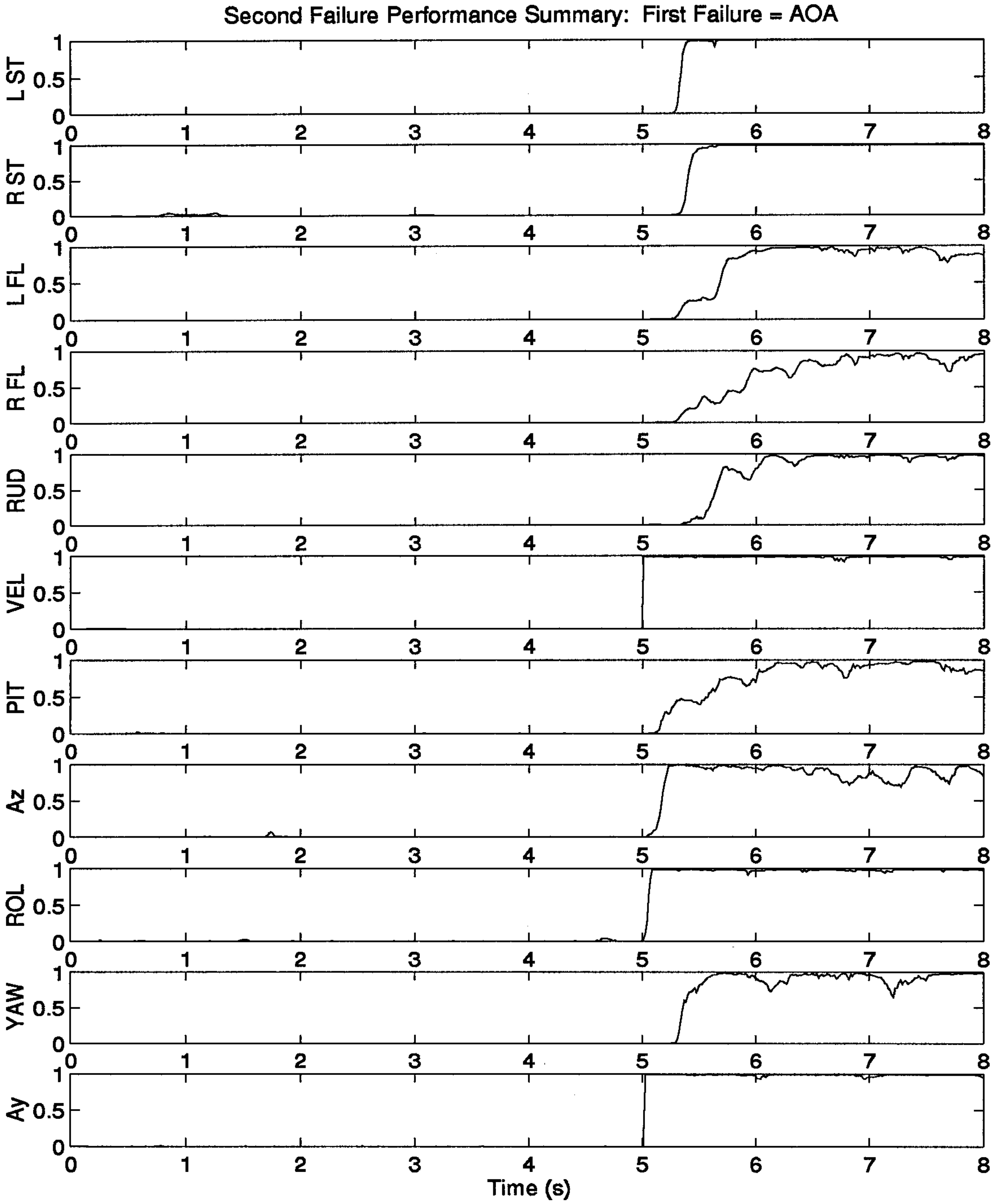


Figure 85. Angle of Attack Sensor Summay Plot - Secondary Failure Induced at 5.0 seconds

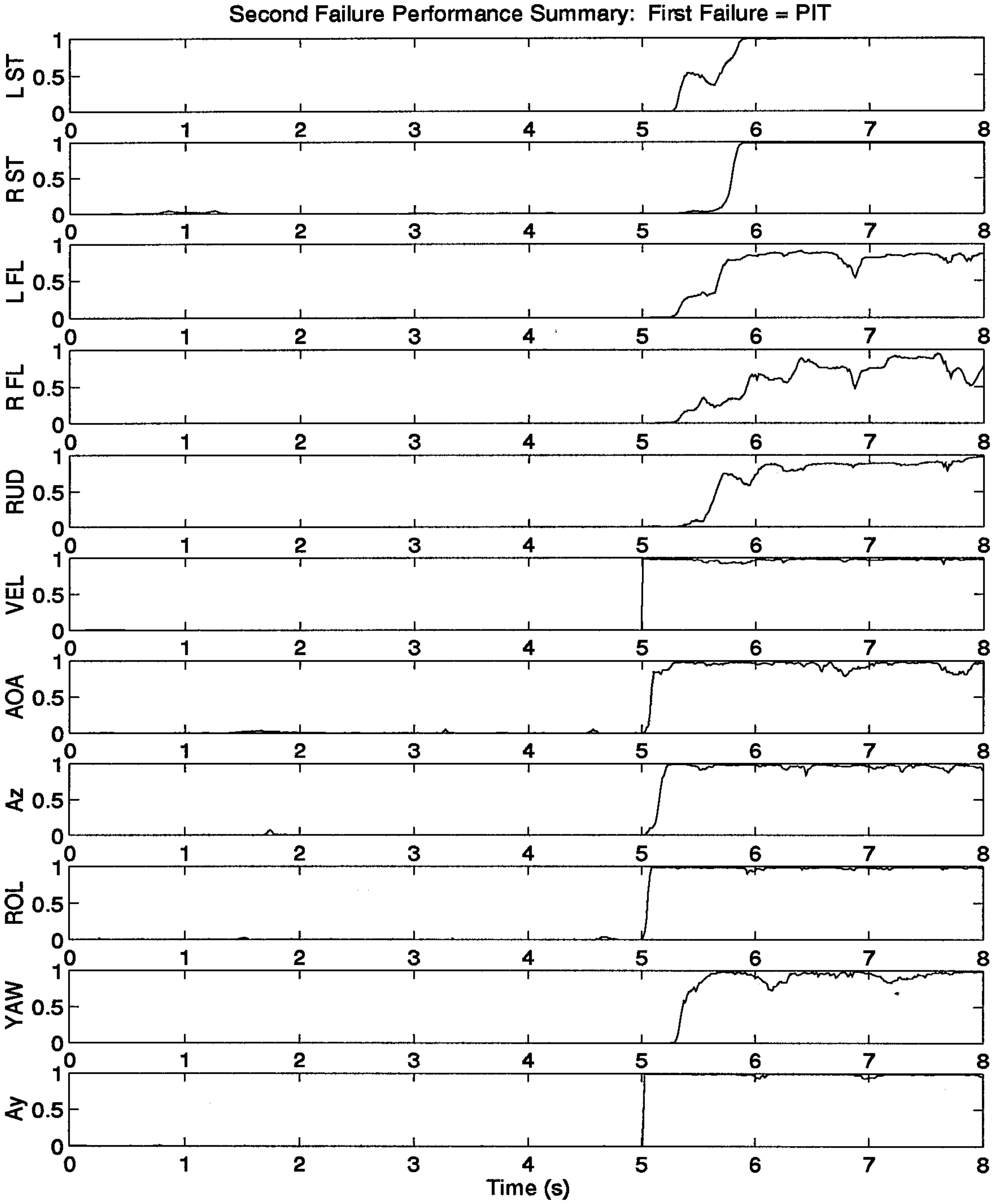


Figure 86. Pitch Rate Sensor Summay Plot - Secondary Failure Induced at 5.0 seconds

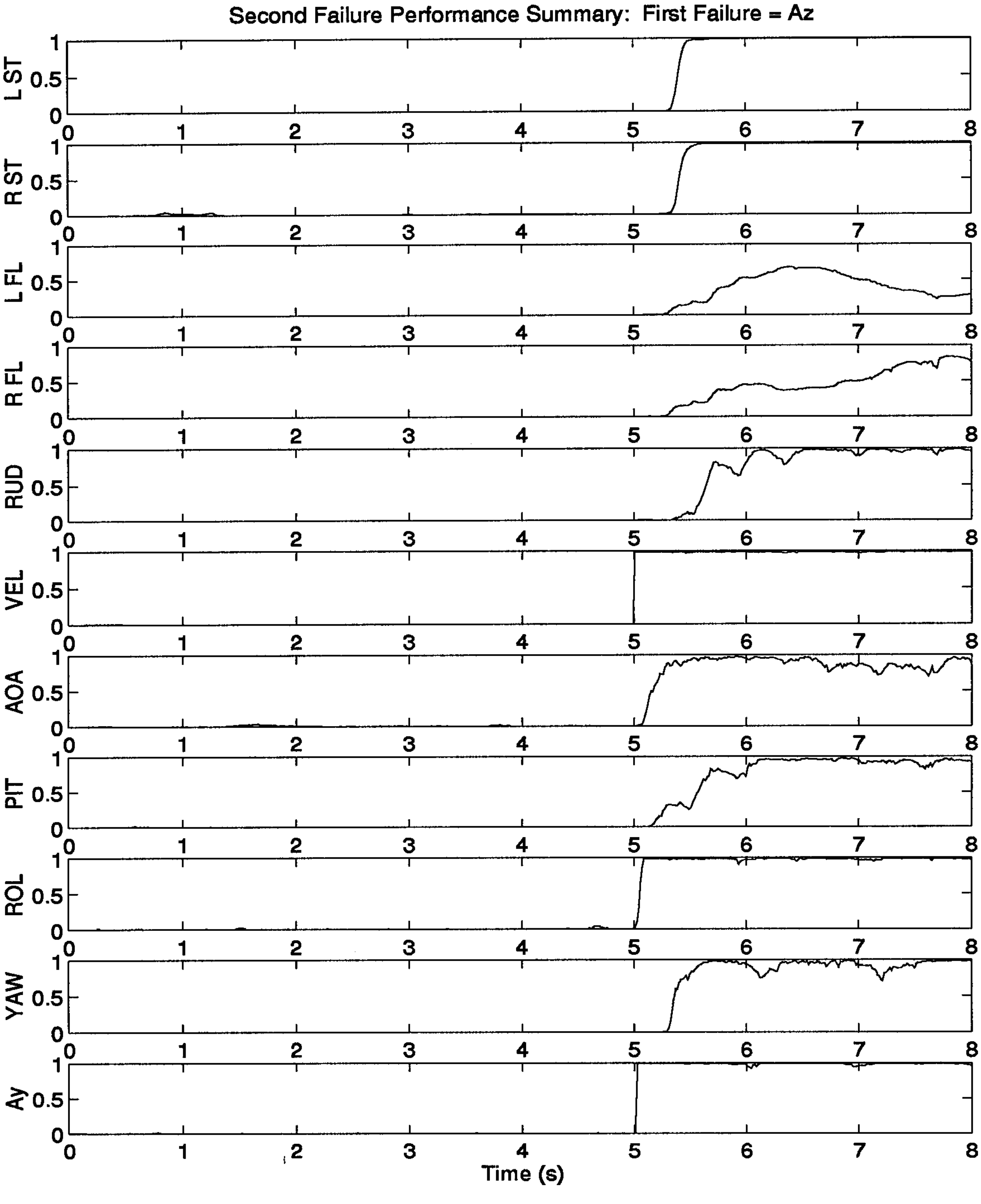


Figure 87. Normal Acceleration Sensor Summay Plot - Secondary Failure Induced at 5.0 seconds

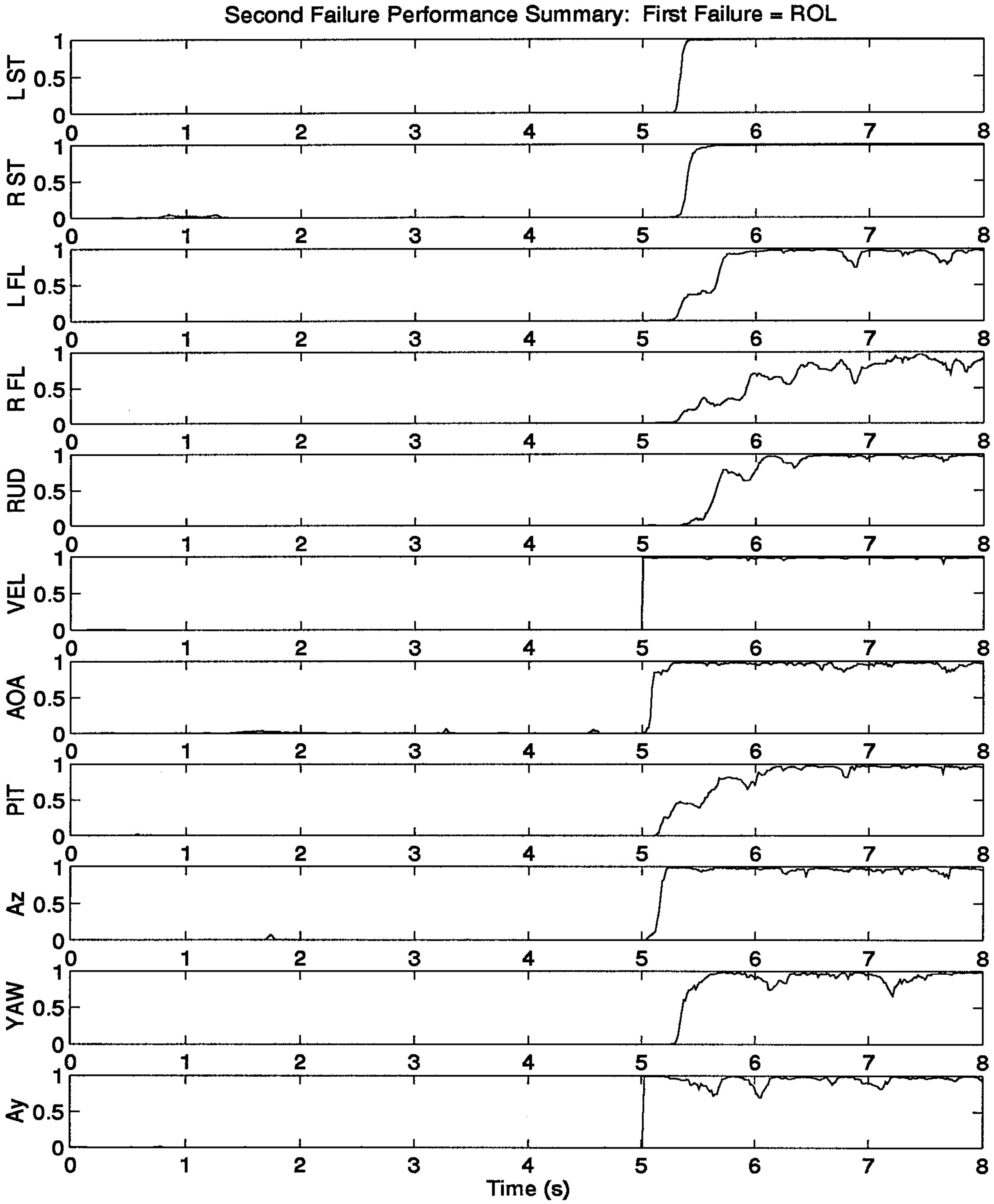


Figure 88. Roll Rate Sensor Summay Plot - Secondary Failure Induced at 5.0 seconds

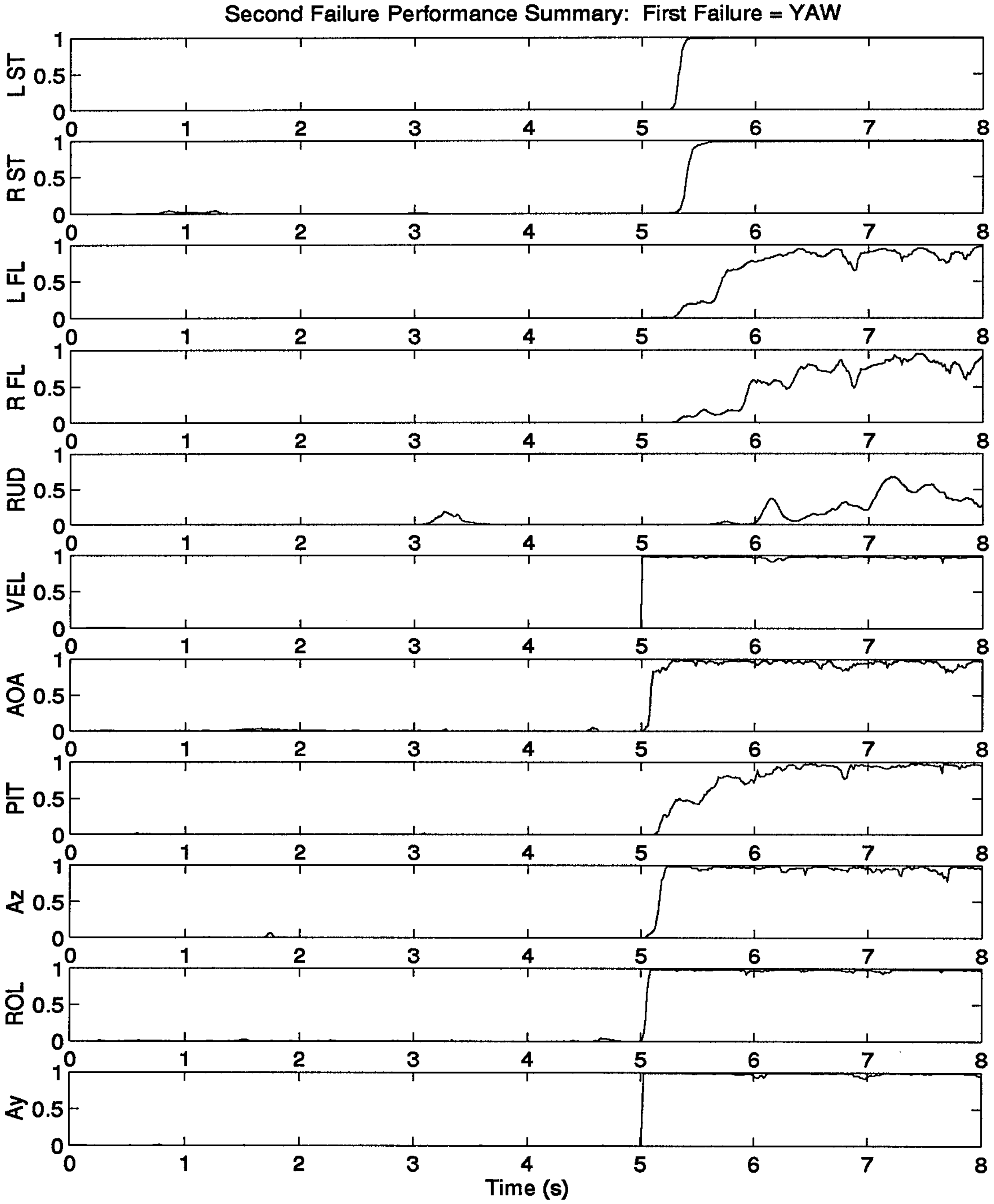


Figure 89. Yaw Rate Sensor Summay Plot - Secondary Failure Induced at 5.0 seconds

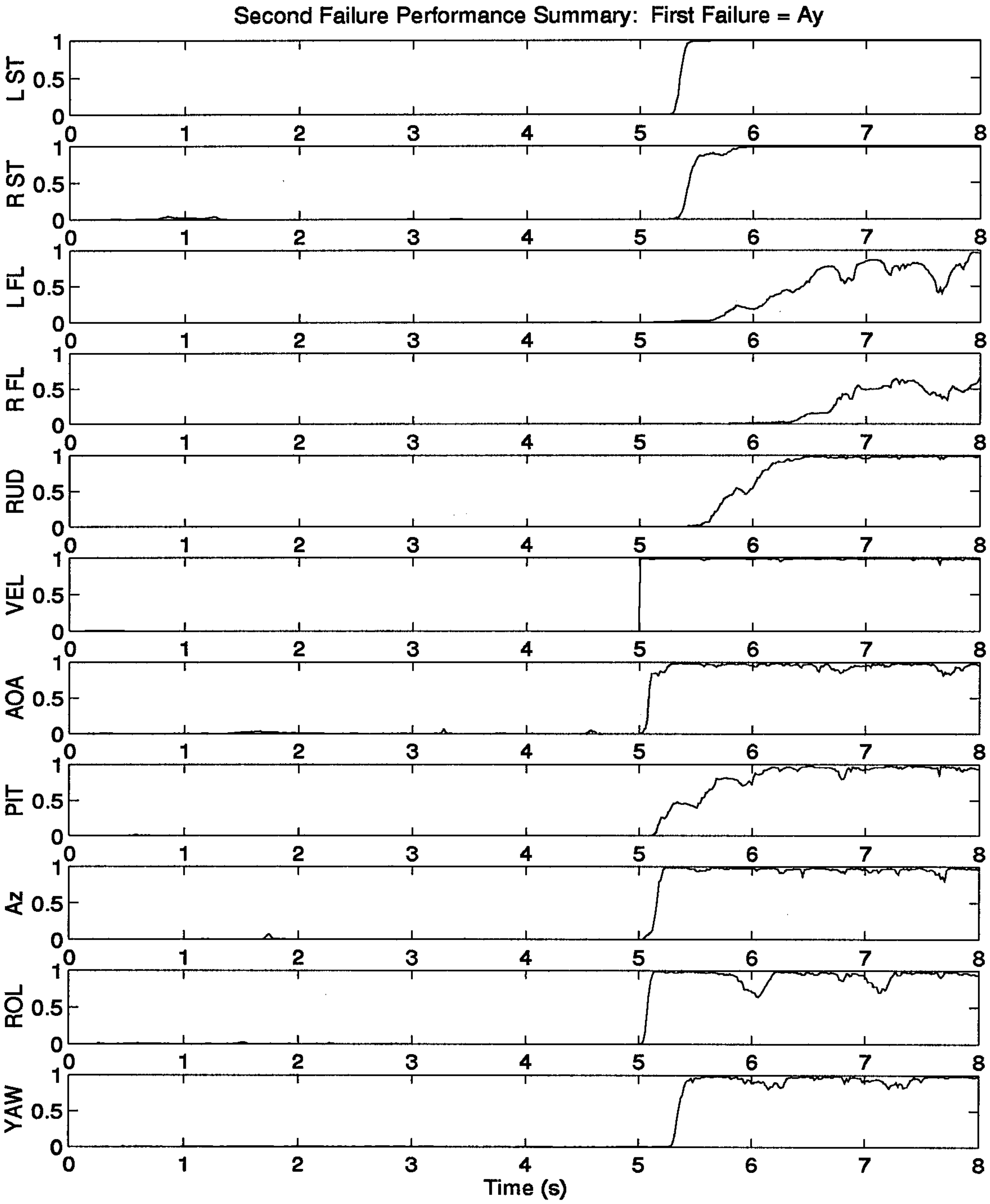


Figure 90. Lateral Acceleration Sensor Summay Plot - Secondary Failure Induced at 5.0 seconds

Bibliography

1. Athans, M., *et al.* "The Stochastic Control of the F-8C Aircraft Using a Multiple Model Adaptive Control (MMAC) Method – Part I: Equilibrium Flight," *IEEE Transactions on Automatic Control*, *AC-22*(5):768–780 (October 1977).
2. Corporation, Northrop. *General Environment for the Simulation of Integrated Systems (GENESIS) Users Manual (PROPRIETARY)* (Version 1.4 Edition), January 1989.
3. Dunn, H. J. and R. C. Montgomery. "A Moving Window Parameter Adaptive Control System for the F8-DFBW Aircraft," *IEEE Transactions on Automatic Control*, *AC-22*(5):788–795 (October 1977).
4. Eckert, S.J. *et al.* "An Application of Nonlinear Filtering to Instrument Failure Detection in a Pressurized Water Reactor," *Nuclear Technology*, (74):139–150 (August 1986).
5. Fitch, Capt James. *Moving-Bank Multiple Model Adaptive Control of a Large Flexible Space Structure with Purposeful Dither for Enhanced Identifiability*. MS Thesis, AFIT/GA/ENG/93D-01, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1993.
6. General Dynamics, Fort Worth Division, "Control Surface Mixer: Dwg 711DFCS007 - Rev A," September 1990.
7. Gustafson, Capt John A. *Control of a Large Flexible Space Structure Using Multiple Model Adaptive Algorithms*. MS Thesis, AFIT/GE/ENG/91D-22, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1991.
8. Haase, A.B. "Slope Data Water-Bottom Multiple Attenuation," *Geophysical Prospecting*, *40*:403–427 (May 1992).
9. Hanlon, Capt Peter D. *Failure Identification Using Multiple Model Adaptive Estimation for the LAMBDA Flight Vehicle*. MS Thesis, AFIT/GE/ENG/92D-19, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1992.
10. Herrera, Theodore D. *Kalman Filter Tracking of a Reflective Target Using Forward Looking Infrared and Doppler Return Measurements*. MS Thesis, AFIT/GE/ENG/91D-25, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1991.
11. Inc., Century Computing. *Simulation/Rapid-Prototyping Facility (SRF) F-16 VISTA Simulation Engineer's Guide*, December 1992.
12. Inc., Century Computing. *Simulation/Rapid-Prototyping Facility (SRF) F-16 VISTA Simulation User's Manual*, October 1992. Draft Sun/UNIX version.
13. Integrated Systems, Inc. MATRIX$_x$ *(registered trademark)* (Version 3.0 Edition), 1992.
14. Lane, Captain David W. *Mutiple Model Adaptive Estimation Applied to the LAMBDA URV for Failure Detection and Identification*. MS Thesis, AFIT/GE/ENG/93D-23, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1993.
15. Leahy, M.B., *et al.* "Multiple Model-Based Control of Robotic Manipulators: An Overview." *Proceedings of the 29th IEEE Converence on Decision and Control (Cat. No. 90CH2917-3)3*. 1976–1977. 1990.
16. Li, X.R., *et al.* "Design of an Interacting Multiple Model Algorithm for Air Traffic Control Tracking," *IEEE Transactions on Control Systems Technology*, *1*:186–194 (September 1993).
17. Magill, D.T. "Optimal Adaptive Estimation of Sample Stochastic Processes," *IEEE Transactions on Automatic Control*, *AC-10*(4):434–439 (October 1965).
18. Martin, Capt Richard Maurice. *LQG Synthesis of Elemental Controllers for AFTI/F-16 Adaptive Flight Control*. MS Thesis, AFIT/GE/ENG/90D-36, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1990.

19. The MathWorks, Inc. *MATLAB (registered trademark)* (Version 4.0a Edition), December 1992.

20. Maybeck, Peter S. *Stochastic Models, Estimation, and Control, I*. Academic Press, Inc., 1979.

21. Maybeck, Peter S. *Stochastic Models, Estimation, and Control, III*. Academic Press, Inc., 1982.

22. Maybeck, Peter S. *Stochastic Models, Estimation, and Control, II*. Academic Press, Inc., 1982.

23. Maybeck, Peter S. and Capt Peter D. Hanlon. "Performance Enhancement of a Multiple Model Adaptive Estimator," *IEEE Conference on Decision and Control* (1993).

24. Maybeck, Peter S. and Capt Donald L. Pogoda. "Multiple Model Adaptive Controller for the STOL F-15 with Sensor/Actuator Failures," *Proceedings of the 28th Conference on Decision and Control*, 1566–1572 (December 1989).

25. Maybeck, Peter S. and Richard D. Stevens. "Reconfigurable Flight Control Via Multiple Model Adaptive Control Methods," *Proceedings of the IEEE National Aerospace and Electronics Conference, Dayton, Ohio*, 441–448 (May 1992).

26. Menke, Timothy E. *Multiple Model Adaptive Estimation Applied to the VISTA F-16 with Actuator and Sensor Failures*. MS Thesis, AFIT/GA/ENG/92J-01, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, June 1992.

27. Menke, Timothy E. and Peter S. Maybeck. "Multiple Model Adaptive Estimation Applied to the VISTA F-16 Flight Control System with Actuator and Sensor Failures," *IEEE Trans. on Aerospace and Electronic Systems, AES-27, No. 3*:470–480 (May 1991).

28. Menke, Timothy E. and Peter S. Maybeck. "Sensor/Actuator Failure Detection in the VISTA F-16 by Multiple Model Adaptive Estimation," *Proceedings of American Control Conference, San Francisco, June 1993* (1993).

29. *MIL-STD-1797A Flying Qualities of Piloted Aircraft*, January 1993.

30. Pogoda, Capt Donald L. *Multiple Model Adaptive Controller for the STOL F-15 with Sensor/Actuator Failures*. MS Thesis, AFIT/GE/ENG/88D-23, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1988.

31. Russell, Capt James E. MS Thesis: in progress, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 199?

32. Stevens, Capt Richard D. *Characterization of a Reconfigurable Multiple Model Adaptive Controller Using A STOL F-15 Model*. MS Thesis, AFIT/GE/ENG/89D-52, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1989.

33. Stratton, Capt Gregory L. *Actuator and Sensor Failure Identification using a Multiple Model Adaptive Technique for the VISTA F-16*. MS Thesis, AFIT/GE/ENG/91D-14, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1991.

34. Sun Microsystems, Inc. *Sun FORTRAN* (Sun release 4.1 Edition), December 1990.

35. Willsky, A. S. "A Survey of Design Methods for Failure Detection in Dynamic Systems," *Automatica*, *12*:601–611 (December 1976).

36. Willsky, A.S. *et al.* "Dynamic Model-Based Techniques for the Detection of Incidents on Freeways," *IEEE Transactions on Automatic Control*, *AC-25*(3):347–359 (June 1980).

37. Yu, C., *et al.* "Multiple-Model Adaptive Predictive Control of Mean Arterial Pressure and Cardiac Output," *IEEE Transactions on Biomedical Engineering*, *39*(8):765–778 (August 1992).

Vita

Captain Peter Keith Eide was born on the fifth of June, 1966, in Menomonee Falls, Wisconsin. It was in this suburb of Milwaukee that he spent the "formative years", not venturing far until higher education called. Then, for four and one half years, he pursued a B.S. in Electrical Engineering and a commission into the U.S. Air Force through ROTC at the University of Wisconsin, Madison. Upon reaching both goals, now 2nd Lt Eide, was assigned as Electronics Warfare Systems Engineer at the Warner Robins Air Logistics Center in Warner Robins, GA. While in this position, he worked many "programs" reserved for lieutenants and in his spare time, brought a $69M Intermediate Level Test Set on line for the AN/ALQ-131 tactical ECM program. Still quite thirsty for knowledge, Capt Eide volunteered and was accepted into the M.S.E.E. program at the Air Force Institute of Technology. While there, emphasis in flight control theory led directly to this research and an accepted follow-on assignment to the Flight Dynamics Laboratory at Wright Patterson AFB. Capt Eide will graduate on December 13th, 1994 and begin work on finding a wife and building a family.

Permanent address: N84W18375 Seneca Ct.
Menomonee Falls, WI 53051

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE December 1994	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE
IMPLEMENTATION AND DEMONSTRATION OF A MULTIPLE MODEL ADAPTIVE ESTIMATION FAILURE DETECTION SYSTEM FOR THE F-16

5. FUNDING NUMBERS

6. AUTHOR(S)
Peter K. Eide

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GE/ENG/94D-06

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Wright Laboratory
WL/FIGS-4, OH 45433-7521

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

ABSTRACT

A Multiple Model Adaptive Estimation (MMAE) algorithm is implemented with the fully nonlinear six-degree-of-motion, Simulation Rapid-Prototyping Facility (SRF) VISTA F-16 software simulation tool. The algorithm is demonstrated to be capable of identifying flight critical aircraft actuator and sensor failures at a low dynamic pressure (20,000 ft, .4 Mach). Research included single and dual complete failures. Tuning methods for accommodating model mismatch, including addition of discrete dynamics pseudonoise and continuous measurement pseudonoise, are discussed and demonstrated. Scalar residuals within each filter are also examined and characterized for possible use as an additional failure declaration voter. Robustness to sensor failures provided by MMAE-based control is also demonstrated. An investigation of algorithm performance off the nominal design conditions is accomplished as a first step towards full flight envelope coverage. The algorithm is composed of a bank of Kalman filters modeled to match particular hypotheses of the real world. Each presumes a single failure in one of the flight critical actuators (left/right stabilators, left/right flaperon, rudder), or sensors (forward velocity, angle of attack, pitch rate, normal acceleration, roll rate, yaw rate, and lateral acceleration), and one presumes no failure. For dual failures, a hierarchical structure is used to keep the number of on-line filters to a minimum. This research advances the technology by testing algorithm performance against the most complete simulation model currently available.

14. SUBJECT TERMS
Multiple Model Adaptive Estimation, MMAE, Kalman Filter, Failure Detection Algorithm, F-16, SRF

15. NUMBER OF PAGES
214

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102